

AD-A055 574 SYSTEM DEVELOPMENT CORP SANTA MONICA CALIF
SOFTWARE ACQUISITION MANAGEMENT GUIDEBOOK; COST EST.--ETC(U)
MAR 78 M FINFER, R MISH F19628-76-C-0236
UNCLASSIFIED SOC-TM-5772/007/02 ESD-TR-78-140

F/G 17/2

N/L

1 of 2
AD
A055574



ESD-TR-78-140

AD A055574

SOFTWARE ACQUISITION MANAGEMENT GUIDEBOOK:
COST ESTIMATION AND MEASUREMENT,

Marsha /Finfer
Russell/Mish
System Development Corporation
2500 Colorado Avenue
Santa Monica, CA 90406



Approved for Public Release;
Distribution Unlimited.

Prepared for

DEPUTY FOR TECHNICAL OPERATIONS
ELECTRONIC SYSTEMS DIVISION
HANSCOM AIR FORCE BASE, MA 01731

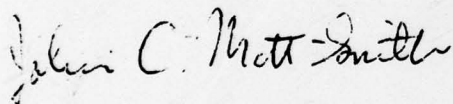
LEGAL NOTICE

When U. S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

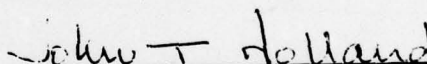
OTHER NOTICES

Do not return this copy. Retain or destroy.

This technical report has been reviewed and is approved for publication.

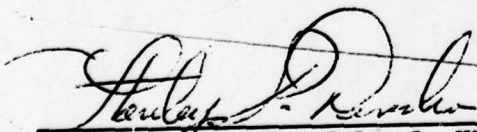


JOHN C. MOTT-SMITH
Project Manager



JOHN T. HOLLAND, Lt Colonel, USAF
Chief, Techniques Engineering Division

FOR THE COMMANDER



STANLEY P. DERESKA, Colonel, USAF
Director, Computer Systems Engineering
Deputy for Technical Operations

NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED
FROM THE BEST COPY FURNISHED US BY
THE SPONSORING AGENCY. ALTHOUGH IT
IS RECOGNIZED THAT CERTAIN PORTIONS
ARE ILLEGIBLE, IT IS BEING RELEASED
IN THE INTEREST OF MAKING AVAILABLE
AS MUCH INFORMATION AS POSSIBLE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-78-140	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Software Acquisition Management Guidebook: Software Cost Estimation and Measurement		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER TM-5772/007/02
7. AUTHOR(s) Marcia F. Finfer Russell K. Mish		8. CONTRACT OR GRANT NUMBER(s) F19628-76-C-0236
9. PERFORMING ORGANIZATION NAME AND ADDRESS System Development Corporation 2500 Colorado Avenue Santa Monica, California 90406		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Command & Management Systems Electronic Systems Division Hanscom AFB, Massachusetts 01731		12. REPORT DATE March 1978
		13. NUMBER OF PAGES 114
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Cost Estimation Cost Proposal Evaluation Cost Proposal Preparation Parametric Models Software Cost Estimation		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Software Cost Estimation and Measurement guidebook is designed to assist Air Force personnel who are responsible for estimating and controlling the costs of embedded software within command, control, and communications systems. It provides a basic understanding of the current methodologies used in the formation of Air Force and contractor software cost estimates. Insight is provided into some of the problems (and reasons for the problems) associated with software cost estimates made by both Government and industry. The guidebook		

over

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

20. (cont'd)

discusses the role of parametric models used in cost estimation and reviews three experimental predictive models. It also discusses the process of monitoring software costs and schedules while providing guidance to relevant military regulations, specifications, standards, and supporting literature. Much of the information and guidance provided is applicable to smaller less complex systems, but in all cases, it should be tailored to the needs of individual projects.

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

PREFACE

This guidebook was prepared by System Development Corporation under the direction of the Computer Systems Engineering Directorate of the Electronic Systems Division (ESD/TOI, formerly MCI), Air Force Systems Command. The Software Cost Estimation and Measurement guidebook is one of a series of Software Acquisition Management guidebooks intended to help ESD Program Office personnel in the acquisition of embedded software for command, control and communications systems. The contents of the guidebook will be revised periodically to reflect changes in software acquisition policies and practices as well as feedback from guidebook users.

The Software Acquisition Management guidebook series is currently planned to cover the following topics (National Technical Information Service accession numbers for those already published are shown in parentheses):

- Regulations, Specifications and Standards* (AD-A016401)
- Contracting for Software Acquisition (AD-A020444)
- Monitoring and Reporting Software Development Status (AD-A016488)
- Statement of Work Preparation (AD-A035924)
- Reviews and Audits
- Computer Program Configuration Management (AD-A047308)
- Computer Program Development Specification (Requirements Specification)
- Software Documentation Requirements (AD-A027051)
- Verification (AD-A048577)
- Validation and Certification
- Overview of the SAM Guidebooks
- Software Maintenance
- Software Quality Assurance (AD-A047318)
- Software Cost Estimation and Measurement
- Software Development and Maintenance Facilities (AD-A038234)
- Life Cycle Events (AD-A037115)

*Revised in March 1978

TABLE OF CONTENTS

	<u>Page</u>
PREFACE	1
LIST OF FIGURES	4
SECTION 1 - INTRODUCTION	7
1.1 Purpose	7
1.2 Scope	7
1.3 Contents	8
SECTION 2 - PROGRAM OFFICE SOFTWARE COST ESTIMATION	9
2.0 Introduction	9
2.1 Program Control	10
2.2 Work Breakdown Structure	16
2.3 Impact of Software Cost Estimation on Contract Cost Type	18
2.4 Summary of a Survey of Program Office Software Cost Estimation Procedures	24
SECTION 3 - OFFEROR'S COST PROPOSAL PREPARATION	29
3.0 Introduction	29
3.1 Cost Estimating Techniques	30
3.1.1 Analogy of Similar Experience	31
3.1.2 Quantitative Method	31
3.1.3 Percent-of-Other-Item Method	32
3.1.4 Rules of Thumb	33
3.1.5 Parametric Equations	33
3.2 Issues Impacting Software Development Costs	36
3.2.1 Complexity of Application	36
3.2.2 Total Software Size	38
3.2.3 Requirements Specification	39
3.2.4 Level of Change in Performance Requirements	41
3.2.5 Documentation Requirements	41
3.2.6 Software Quality Requirements	42
3.2.7 Software Development Schedule	42
3.2.8 Type of Software Development Effort	45
3.2.9 Personnel Requirements	45
3.2.10 Development Methodology	46
3.3 Project Management and Scheduling Plans	47
3.3.1 Task Segmentation/WBS Definition	50
3.3.2 Scheduling of WBS Elements	51
SECTION 4 - ROLE OF PARAMETRIC MODELS	53
4.0 Introduction	53
4.1 Parametric Models	53
4.2 Models Versus Methods in Cost Estimation	54
4.3 Development of Cost Estimation Relationships (CERs)	55

TABLE OF CONTENTS (cont'd)

	<u>Page</u>
SECTION 5 - COST PROPOSAL EVALUATION	59
5.0 Introduction	59
5.1 The Mechanics of Evaluation	59
5.2 Cost Analysis of Technical Activities	60
5.3 Cost Analysis of Technical/Financial Monitoring Activities	64
APPENDIX A - REFERENCES	67
APPENDIX B - SOFTWARE COST ESTIMATION MODELS	71
1.0 Introduction	71
1.1 Hahn & Stone Software Transfer Cost Estimation Technique	71
1.1.1 Factors Considered in the Hahn & Stone Model	71
1.1.2 Hahn and Stone Model	76
1.1.3 Summary Evaluation of Hahn & Stone Software Transfer Cost Estimation Technique	83
1.2 Putnam General Solution to the Software Sizing and Estimating Problem	86
1.2.1 Factors Considered in the Putnam Model	86
1.2.2 Putnam Life Cycle Model	86
1.2.3 Evaluation of the Putnam Model	91
1.3 The Tecolote Provisional Model for Estimating Computer Program Development Costs	93
1.3.1 Factors Considered in the Tecolote Model.	93
1.3.2 Tecolote's Provisional Cost Model	95
1.3.3 Evaluation of Tecolote Approach	101
APPENDIX C - GLOSSARY	107
APPENDIX D - OFFICIAL GOVERNMENT DOCUMENTS FOR COST ESTIMATION AND MEASUREMENT	113

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
Figure 1. Model System Acquisition Process	1
Figure 2. Typical Program Office	10
Figure 3. Phased Software Cost Estimation	13
Figure 4. Work Breakdown Structure	18
Figure 5. Type of Contract vs Degree of Risk	20
Figure 6. Contract Type vs Risk Considerations for Software	21

LIST OF FIGURES

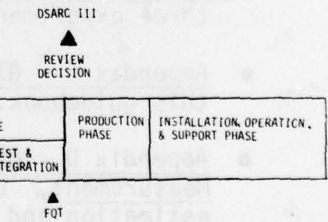
<u>Figure</u>	<u>Page</u>
Figure 7. ASPR References Regarding Contract Type Selection	23
Figure 8. Survey of ESD Program Office Software Cost Estimation Methods	26
Figure 9. Rules of Thumb	34
Figure 10. Estimated Distribution of Resources for a Medium- Large Project	44
Figure 11. Examples of Gantt, Project Network, and Time-Scaled Network Charts	49
Figure 12. Software Transfer Considerations	73
Figure 13. Manual Transfer Tasks	77
Figure 14. Conversion Production Rates (Instructions or State- ments/Man-Day)	82
Figure 15. Development Production Rates (Instructions/Man-Day) . . .	83
Figure 16. Division of Effort for Complete Redesign Task	84
Figure 17. Documentation Degradation Factors	84
Figure 18. Program Instability Degradation Factors	85
Figure 19. Project Profile	87
Figure 20. Typical CSC Application of Manpower to a Software Development Project	87
Figure 21. Life Cycle Integral and Derivative Curve Forms	88
Figure 22. Application Software Life Cycle	89
Figure 23. Programming Rate Versus Difficulty	91
Figure 24. Basic Data for Tecolote Provisional Model	96
Figure 25. Fast Storage vs Targets Tracked	97
Figure 26. Operating Instructions vs Fast Storage	99
Figure 27. Delivered Instructions vs Operating Instructions	99
Figure 28. Man-Months vs Operating Instructions	100
Figure 29. Man-Months vs Delivered Instructions	100
Figure 30. Inferred per Man-Month Factors	101
Figure 31. Summary of Provisional Software Estimating Relation- ships	103
Figure 32. Adjustment Factor for Different Cost Factor Assumptions .	104
Figure 33. Provisional Software Development Cost Model	105

is designed to assist
and controlling the
communications
pecifically towards
ber of the Engineering
generally responsible
tion and guidance
but in all cases,
ts

ce 800 series

basic understanding of Force and contractor of the problems (and estimates made by parametric models use active models. It also schedules while providing standards, and

strategy impacts total
ies required by both
allocation, and expen-
lebook is based on the
1, unless otherwise



process*

1.3 CONTENTS

The subsequent contents of this guidebook consist of four sections and three appendixes, as follows:

- Section 2 - Program Office Software Cost Estimation. Discusses those factors associated with the PO that directly contribute to the formation of the system cost estimate, with particular emphasis on the software element portion of that estimate. The topics covered run from the end of the Conceptual Phase and subsequent program decision to submission of the RFP for Full-Scale Development, including PO Program Control, the Work Breakdown Structure, contract cost type, and PO software cost estimation procedures.
- Section 3 - Offeror's Cost Proposal Preparation. Concentrates on the various offeror tasks associated with preparing a software cost proposal in response to an RFP for Full-Scale Development. This section presents an overview of the numerous software cost estimating techniques used, the project-dependent factors that impact software development costs and analyses, and task specification and scheduling with regard to the WBS.
- Section 4 - Role of Parametric Models. Discusses the use of parametric models in software cost estimation. Provides information regarding the development of parametric models. Discusses their strengths and weaknesses.
- Section 5 - Cost Proposal Evaluation. Provides information for use by the PO in offeror cost proposal evaluation. It also discusses issues relating to the requirements and procedures for cost reporting and performance measurement of command, control, and communication system acquisition during Full-Scale Development.
- Appendix A - References. Presents a list of numbered references.
- Appendix B - Software Cost Estimation Models. Discusses and evaluates three experimental software estimation models.
- Appendix C - Glossary. Defines specific terms and acronyms used in this guidebook.
- Appendix D - Official Government Documents for Cost Estimation and Measurement. Lists Government documents that impact software cost estimation and measurement.

SECTION 2 - PROGRAM OFFICE SOFTWARE COST ESTIMATION

2.0 INTRODUCTION

In a typical system development effort, the Required Operational Capability (ROC)* is analyzed to determine if a solution is feasible and affordable within current budget and technology constraints. The former is determined through the medium of Life Cycle Costing (LCC). LCC is the Government's total cost of owning a system, subsystem, or component over its full life. It includes development, production, operation, and support costs. From a system point of view, LCC estimates are required in support of program decisions.

While program decision-making is a continuing process throughout the system's life cycle, the groundwork is laid during the Conceptual Phase leading to the first decision to proceed into the Validation Phase, made by Defense System Acquisition Review Council (DSARC) for major systems.

By the end of the Conceptual Phase,** an acquisition strategy is developed (by the PO) which is compatible with the program's preliminary performance, schedule, and projected costs. This strategy is developed, in part, by obtaining and evaluating system descriptions, ground rules, constraints, and assumptions upon which the cost estimates are based. In addition, available historical data is analyzed to substantiate the projected costs of the new system.

During the Validation Phase, performance, schedule, and estimated costs are further validated and refined. The software element and its functions are more clearly identified to provide a more definitive basis from which to proceed with the software cost estimating process. The major objective of the Validation Phase is to assure that the system chosen for Full-Scale Development is both technically and economically feasible. Another result is a better definition of the program characteristics pertaining to system performance, cost, and schedules. This section discusses the refinement of the software cost estimate during the Validation Phase.

*Recently replaced by General Operational Requirement (GOR).

**The Conceptual Phase generally terminates with DSARC milestone 1. DSARC 1 permits the Secretary of Defense to endorse or redirect a major weapons system. A Decision Coordination Paper (DCP) is prepared to support DSARC reviews, and contains system information regarding operational needs, system performance, and associated program cost data. Whenever a DSARC is not required, similar reviews are held at the Air Force level.

In the same way that a procurement strategy may not rigidly adhere to the optimum model, the specializations within acquisition management may not be consistent for any given program's acquisition. The discussion presented in 2.1 describes the generic activities conducted by the discipline of Program Control with emphasis on those activities closely associated with derivation of the software cost estimate. The activities of Program Control are brought together by the Work Breakdown Structure (WBS) (see 2.2) which describes the relationship of system costs, tasks, and products. The determination of contract cost type in relationship to the degree of risk in the system is discussed in 2.3. A synopsis of PO activities concerned with derivation of the software cost estimate is presented in 2.4.

2.1 PROGRAM CONTROL

Program Control is a functional organization within the PO which is charged with the business operations necessary to the acquisition of a system. It is responsible for the system cost estimate, including developing, monitoring, and assessing schedules and funds for the program at various stages of its life cycle. This activity includes obtaining software cost estimates which are used to evaluate the cost effectiveness of system tradeoffs and contractor cost proposals for the Full-Scale Development Phase. These activities result in the preparation and maintenance of the schedule and financial requirements data contained in the Program Management Plan (PMP), DCP, and other system documentation. The contents of the system cost estimate and its formulation, however, depend to some extent on the specific type of program and acquisition strategy.

The relationship of Program Control to the typical PO is shown in Figure 2. Program Control may be organized by function (i.e., financial management and discipline management). Its organizational structure is determined by the complexity of the program and the prerogatives of the Program Manager (PM). The specific responsibilities of Program Control are not altered by its internal organization. A more detailed description of the organization and functions of Program Control is presented in Chapter 6 of AFSCP 800-3.

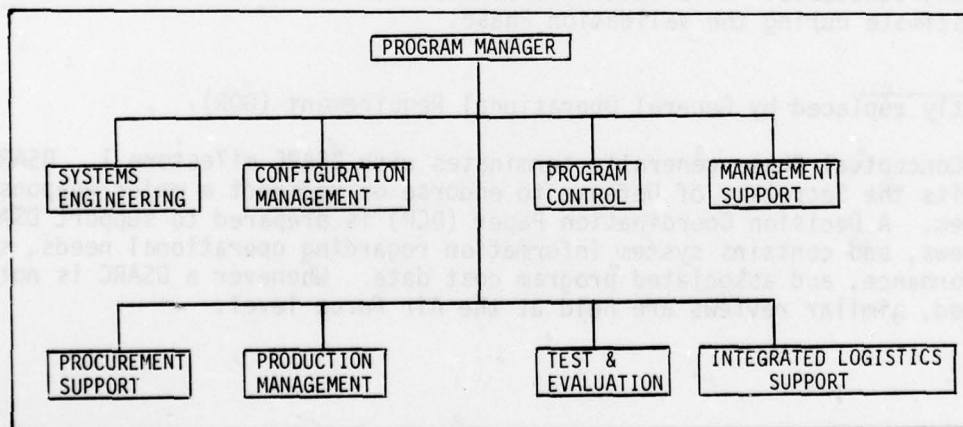


Figure 2. Typical Program Office (see Ref. [2])

Preparation of the software cost estimate should be an iterative process. As the program advances in time and detail (i.e., system requirements are defined and validated, alternatives and tradeoffs are studied and resolved, and a feasible approach is ascertained), additional and more definitive information becomes available which is necessary for obtaining realistic cost-to-complete or total cost estimates.

AFSCM 173-1 provides a comprehensive guide for cost estimating in support of system acquisition activities. In Ref. [1], Doty (1) recommends points in the life cycle when estimates should be made, (2) discusses algorithms for estimating software development resources and time requirements, (3) points out techniques for assessing the feasibility of the proposed program, and (4) describes pitfalls to be avoided in making use of the cost estimates in program management.

Since the process of estimating software development resource and time requirements is a complex task requiring in-depth knowledge of each program, there is no simple approach that can be given in this, or any other guidebook, which will guarantee valid software cost estimates. The following steps are given as a general guide to the tasks necessary to prepare the cost estimate, and should be used in conjunction with other Air Force guidance, industry guidelines, and experienced personnel:

- Define the Objectives Of the Software Estimation Task. This task includes the detailed planning required to support the preparation of the software cost estimate, including determining the applicable items of the program WBS, the method of software acquisition, the information necessary to support definition of software requirements, and schedule constraints.
- Define Personnel, Resources, and Time Requirements Needed For the Software Cost Estimation Process. This task includes identifying the team of people responsible for making the software cost estimate, as well as allocating adequate resources for planning and coordinating.
- Describe the Software Requirements. Initially, this task will be directed at scoping the general software requirements. However, as more system information becomes available, software requirements will become less ambiguous and more detailed. This task should also include documenting software-related assumptions made and definitions derived basic to the cost estimate. The level of detail provided in the description of software requirements is dependent on the objectives established in the initial task. Therefore, it is important that the initial planning activity provide for adequate resources (manpower and time) to allow for accumulation and refinement of system information necessary for software requirements analysis.

- Identify the Software Cost Estimation Techniques To be Used. Confidence can be increased in the resultant software estimate if several different techniques are used during the iterative process of cost estimation preparation. For example, modeling techniques should always be augmented by analysis of historical data obtained from analogous systems. Expert opinion obtained from more than one expert will temper subjective bias or insufficient recall. (Software cost estimation techniques are described in more detail in Sections 3 and 4.)
- Compare and Refine the Derived Software Cost Estimates. This task should include resolving inconsistencies and ambiguities in the accumulated data. Care should be taken to generate a conservative, but realistic, cost estimate. Rules-of-thumb or historical cost data should be used for comparisons of functionally similar systems in the refinement process. Program-dependent characteristics, such as schedule constraints, concurrent hardware development, and completeness of software requirements, should be assessed to determine incompatibilities or conflicts which may perturb or alter assumptions upon which the software cost estimate was based. In addition, this task should examine the estimate for completeness; that is, it should include provisions for both Government in-house costs and contractor development costs.

Because of the difficulty of accurately estimating software development costs, especially at points in the system acquisition life cycle where adequate technical information is not yet available, an iterative cost estimation process is the only mechanism by which the Government can expect to obtain reasonably valid cost estimates.

Cost estimates may be prepared at any point in the acquisition cycle, but it is important to have an estimate at the following decision points:

- Program Decision (between Conceptual and Validation Phases)
- Ratification Decision (between Validation and Full-Scale Development Phases)
- Production Decision (between Full-Scale Development and Production Phases)

Figure 3 depicts summary information for developing estimates of computer software costs within the system acquisition framework. The data presented in Figure 3 demonstrates that as more information becomes available to the software cost estimator, some of the ambiguities that initially exist in the formation of the software cost estimate are resolved. The consequence is an increase in confidence of the estimate.

ESTIMATE	USE	REQUIRED TECHNICAL INPUTS	REQUIRED FINANCIAL INPUTS	SIZING BASIS	
① Initial Program Budgetary Estimate (Conceptual Phase)	<ul style="list-style-type: none"> To formulate Life Cycle Costs and Design-to-Cost thresholds 	<ul style="list-style-type: none"> Conceptual System Definition Initial Software Sizing Estimate RFI Feasibility & Risk Assessment 	<ul style="list-style-type: none"> POM 	<ul style="list-style-type: none"> Total number of object instructions 	<ul style="list-style-type: none">
② Independent Validation Cost Estimate (Validation Phase- prior to RFP Release, or Program Decision)	<ul style="list-style-type: none"> To assess program feasibility; used in: <ul style="list-style-type: none"> - POM - APP - DCP/PM - Resources Annex 	<ul style="list-style-type: none"> WBS Performance Specification SRR Results SOW Source Selection Plan RFP GFI/GFM CDRL CRISP 	<ul style="list-style-type: none"> POM DCP/PM Resources Annex Advanced Procurement Plan 	<ul style="list-style-type: none"> Total number executable object instructions minus data areas 	<ul style="list-style-type: none">
③ Independent Full Scale Development (FSD) Cost Estimate (Ratification Decision Phase)	<ul style="list-style-type: none"> To evaluate the contractors' proposed costs 	<ul style="list-style-type: none"> Hardware/Software/Firmware Tradeoffs Feasibility and Risk Assessment SDR(s) Results 	<ul style="list-style-type: none"> DCP/PM Updates POM Evaluated cost and schedule control system proposed thresholds 	<ul style="list-style-type: none"> Total number executable object instructions minus data areas for reusable code 	<ul style="list-style-type: none">
④ Update of FSD Cost Estimate (From Preliminary Design Review to remainder of development)	<ul style="list-style-type: none"> To update and monitor development costs 	<ul style="list-style-type: none"> PDR Results CDR Results ECP Review CPCI Testing Review FCA PCA FQR 	<ul style="list-style-type: none"> DCP/PM Updates SAR (if required) POM Approve Cost/Schedule Control System Thresholds Monitoring of Cost/Schedule ECP Cost Analysis Monthly estimates of cost to complete 	<ul style="list-style-type: none"> Total number source code instructions 	<ul style="list-style-type: none">

LEGEND

APP - Advanced Procurement Plan
 CDR - Critical Design Review
 CDRL - Contract Data Requirements List
 CRISP - Computer Resources Integrated Support Plan
 DCP/PM - Decision Coordinating Papers/Program Memorandum
 ECP - Engineering Change Proposal
 GFI/GFM - Government Furnished Information/Government Furnished Material
 PDR - Preliminary Design Review
 POM - Program Objectives Memorandum
 RFI - Request for Information

RFP - Request for Proposal
 SAR - Selected Acquisition Review
 SDR - System Design Review
 SRR - System Requirements Review
 WBS - Work Breakdown Structure
 SOW - Statement of Work

A

	SIZING BASIS	% ERROR	COST ESTIMATORS TO GOVERNMENT AND INDUSTRY	OTHER COMMENTS
	<ul style="list-style-type: none"> Total number of object instructions 	<ul style="list-style-type: none"> Up to 200% sizing error 62% resource estimate error 	<ul style="list-style-type: none"> Government: <ul style="list-style-type: none"> for Conceptual Definition 0-7 people 0-168 man/months for Program Decision 3-7 people 9-63 man/months Contractor <ul style="list-style-type: none"> generally none 	<ul style="list-style-type: none"> A life-cycle cost estimate Industrial input may be used either informally or through an KFI.
Plan	<ul style="list-style-type: none"> Total number executable object instructions minus data areas 	<ul style="list-style-type: none"> Up to 100% sizing error 62% resource estimate error 	<ul style="list-style-type: none"> Government: <ul style="list-style-type: none"> 10-20% of contractors' FSD Costs Contractor <ul style="list-style-type: none"> 10-20% of contractors' FSD Costs for each contractor used 	<ul style="list-style-type: none"> Independent cost estimates for each approach need to be developed if competing contractors' approaches are widely divergent. Design-to-Cost targets to sub-program level (if applicable) need to be developed. LCC estimates are also updated.
Schedule and	<ul style="list-style-type: none"> Total number executable object instructions minus data areas for reusable code 	<ul style="list-style-type: none"> Up to 75% sizing error 62% resource estimate error 	<ul style="list-style-type: none"> Government <ul style="list-style-type: none"> 20-40% of contractor's FSD costs Contractors (see Figure 3 of Ref. [1]). 	None
Control Schedule Cost	<ul style="list-style-type: none"> Total number source code instructions 	<ul style="list-style-type: none"> Up to 50% sizing error, improving to zero at project completion 51% resource estimate error 		<ul style="list-style-type: none"> Monthly Estimated Actual Costs will be made and compared with contractor estimates.

Figure 3. Phased Software Cost Estimates
(Adapted from Ref. [1])

To support major program decisions or to reduce uncertainties that may exist with respect to program elements, the PO can obtain an Independent Cost Estimate (ICE)* from an independent Government team (e.g., the ESD Cost Analysis Division) or other outside agency (e.g., the MITRE Corporation, a Federal Contract Research Center). Generally, the independent estimator uses a parametric equation based on an established cost estimating relationship in which the projected size of the software is a key element. Because the PO generally provides the estimator with the projected software size, the independence of the ICE is questionable because it is based on the same parameter as the estimate generated by Program Control. The usefulness of the ICE will be improved if the independent Government team derives its own estimate of software size because of the extreme importance of this parameter to cost estimates.** The PO has the final responsibility for accepting or rejecting the ICE.

Program Control is also responsible for cost and schedule status reporting for approved programs. This task includes analyzing data to determine contractual financial progress as well as maintaining information concerning the current adjusted target cost/price and adjusted ceiling price.

Another Program Control responsibility includes tracking costs associated with program changes. This may include coordinating plans, changes, or modifications within the program to achieve cost and schedule goals specific to the program. The number and size of changes [i.e., Engineering Change Proposals (ECPs)] for any given system generally cannot be determined in the early stages of program planning and software cost estimate preparation. Clearly, ECPs should be expected in all system acquisitions. Further, they will impact total system cost. A large ECP may require a new cost estimate to be made to support evaluation of the ECP submission if it requests new system functional or performance requirements. In addition, program cost change histories must be maintained to reflect contract performance in terms of actual costs, estimated costs, and changes in system functional/performance requirements.

Program Control's program cost/schedule management responsibilities are brought together through the selection and use of the WBS. The WBS provides a framework for the Government's cost estimate, the contractor's response to the RFP, and the contract cost collection and reporting system. Because this vehicle is an important tool to many Program Control functions, it is described in more detail in 2.2. In addition, a discussion of some of the problems in applying the WBS to system acquisitions is presented in 5.3.

*An ICE represents an estimate of costs to be paid by the Government for a program segment. The scope of the estimate depends on the purpose of the ICE, and may include costs for the entire system or some program element (e.g., software).

**Actual size estimates have been observed to exceed estimated size estimates by 200-300 per cent.

2.2 WORK BREAKDOWN STRUCTURE

A WBS is a hierarchical and graphical representation of the tasks and products that comprise a system acquisition, e.g., "a product-oriented family tree which completely defines the project/program. A WBS displays and defines the product(s) to be developed or produced and relates the elements of work to be accomplished to each other and to the end product" (see Ref. [3]). The specific role of the WBS in software cost estimation and measurement is to provide a basic framework and coordination point for planning, technical management, resource allocation, and cost estimates. The WBS requires that each deliverable product's costs be visible for efficient contract control*. It also provides a basis for auditing each offeror's management control system after contract award. Each element in a WBS level should represent non-overlapping subdivisions of products or tasks. A more detailed discussion of the WBS is presented in Appendix A of Ref. [4]. See also MIL-STD-881A.

The upper levels of the hierarchical structure of the WBS (as prescribed in MIL-STD-881A) are defined as follows:

- Level 1 represents the entire defense materiel item (i.e., the overall collection of tasks and products of the system).
- Level 2 represents major elements of the system (e.g., an aggregation of services).
- Level 3 represents subordinate elements to Level 2 (e.g., a type of service).

The number of levels specified in a WBS for a given system depends upon the type of system acquired and the level of work the Government desires to monitor and control. A primary objective of the WBS is to derive the contracted work effort into manageable units. Depending upon the size and complexity of a specific task, the WBS may be extended to many lower levels to reflect how the work is to be accomplished. In structuring the lower levels of a WBS, the work effort is generally categorized as follows:

*However, not all software developed for a specific system is visible for contract control. Support software (e.g., compilers, and Program Support Libraries) is often not a contractually-specified CPCI. Since applications software may incur severe slippages when support software development is delayed, there may be no indication of the problem until late in the program due to the lack of visibility into the development progress of non-deliverable software. In addition, total software development costs for each given system cannot be accumulated and achieved if support software is non-deliverable. It would appear that for both contract control and accuracy in historical cost data recording, support software should be a contractually specified CPCI.

- Individual tasks resulting in a specific end product (i.e., work packages).
- Support effort or work not resulting in a final product (i.e., level of effort).
- Factored effort directly related to other identified tasks (i.e., apportioned effort, such as quality assurance).

The Contract Work Breakdown Structure (CWBS) is the complete WBS for a specific contractor. It comprises the selected project summary WBS elements included in the contract work statement. The individual contractor can extend the CWBS to lower levels to reflect the way in which his work is to be performed. As such, the CWBS should:

- Display and define the products to be developed and the services to be performed.
- Relate the tasks to be accomplished and their relationship to each other and to the end product.
- Allow for the unique identification of each task which may reflect the overall hierarchy of project tasks, and provide a mechanism for cost accounting for each partitioned task.

A preliminary CWBS is derived by selecting a subset of the project summary WBS and extending the elements to a level lower than Level 3. A preliminary CWBS must be included in each RFP for every contract planned (see Figure 4).

The WBS is used throughout the system acquisition cycle. Initially, the preliminary project summary WBS is used by the Department of Defense (DoD) to define Government tasks and products, appropriate contractor tasks and products, and to support program approval. During the Validation Phase, the WBS forms a framework for preparation of the Statement of Work (SOW), although the SOW generally defines the details of tasks to a greater specificity than the preliminary WBS. The use of the WBS, in conjunction with the Full-Scale Development Phase cost accounting procedures described in Section 5, allows for control over the reporting of contractor technical status and resource expenditures. The WBS can also provide the mechanism for acquiring historical data in a cost element data base. A data base of this nature will provide qualitative values that would enable PO personnel to realistically price types of software (e.g., support software or data base management systems) with current and comparable data obtained from analogous systems/subsystems.

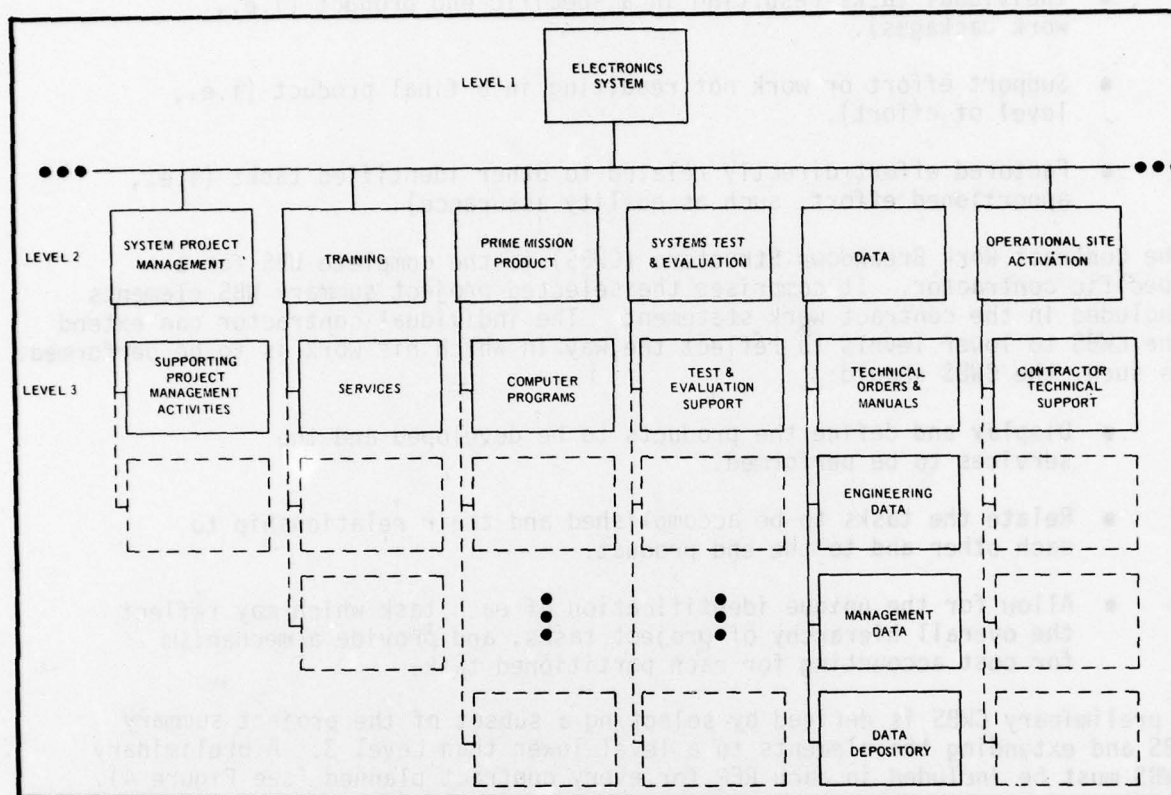


Figure 4. Work Breakdown Structure (see Ref. [3]).

2.3 IMPACT OF SOFTWARE COST ESTIMATION ON CONTRACT COST TYPE

The level of confidence in the software cost estimate should directly impact the contract cost type selected. Although there are no unique contract types for software development, the type of contract selected for the acquisition of software has a direct effect upon the types and amounts of information needed by the PO for cost estimation and contractor selection. The quality and quantity of information available for software cost estimation directly impacts the level of confidence in the estimate. Further, confidence in the estimate is reduced if the information on which it is based (i.e., the functional/performance requirements) is imprecise or is likely to change between the time of procurement and completion of software development.

Cost Reimbursement contracts should almost always be used for Validation Phase contracts. Full-Scale Development contracts range from Cost Reimbursement to Fixed Price. Such contracts include tailored variations in incentives and awards related to fee or profit determination. The selection of contract type for software depends upon the following considerations:

- Procurements which do not yet have approved or authenticated Computer Program Development (Part I) Specifications should normally have Cost Reimbursement contracts. This allows both the Government and the contractor some latitude to make timely accommodations to changing requirements.
- Procurements based upon approved or authenticated Development Specifications may have Fixed Price contracts. However, this type of contract usually makes it difficult for changes in performance requirements, costs, or schedules to be accommodated by either the Government or the contractor in a timely manner.
- Procurements for Full-Scale Development of software usually involve changing engineering requirements because of changing mission and performance requirements. These changes often occur as hardware and software development proceeds concurrently. It is frequently cost effective to correct system functional/performance oversights in the software rather than the hardware. This type of software change may occur as late as System Development Test and Evaluation (DT&E) when the system is checked out. To allow both the Government and contractors to more flexibly incorporate required changes, a Cost Reimbursement contract is recommended for most Full-Scale Development contracts. Changes to Fixed Price contracts can also be handled through Fixed Price ECPs.
- Procurements for off-the-shelf software are often appropriate for Fixed Price contracts, provided the steps to qualify the software are clearly understood.

The final selection of a contract type should result in one which will give assurance of contract performance in a manner most advantageous to the Government, having considered and allowed for reasonable contractor technical and cost risks. Figure 5 depicts summary information on the degree of risk and type of contract selected. Figure 6 presents a more detailed presentation of the contract type and risk consideration.

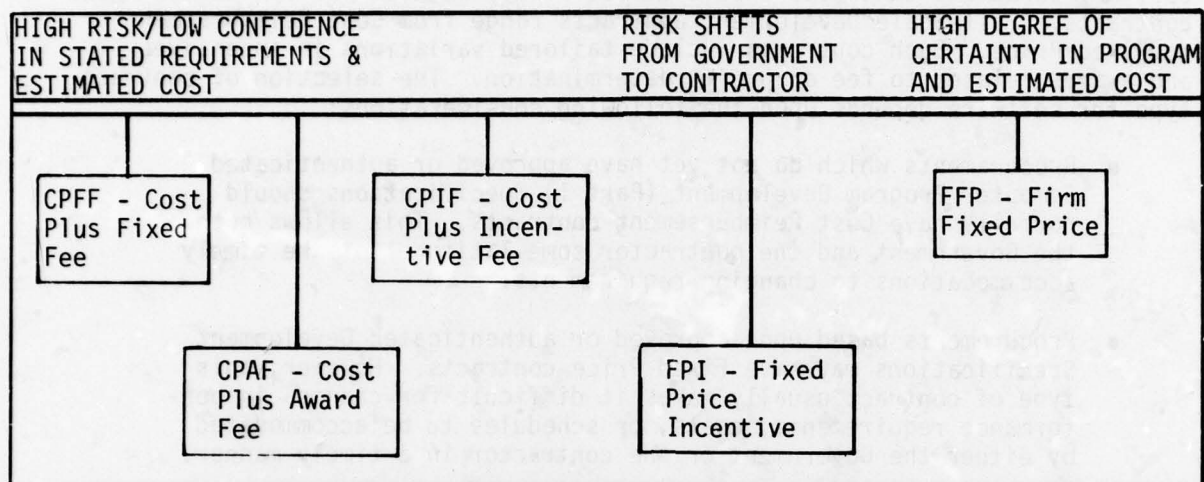


Figure 5. Type of Contract vs Degree of Risk

Figure 7 presents an abbreviated list of pertinent ASPR references. These references provide and lead to definitive guidance in determining and selecting an appropriate type of contract in the best interests of the Government and one which will provide a contractor with a reasonable hedge against technical and cost risks. The risks are related to the confidence in the job definitions as documented in specifications and in the SOW as well as to the confidence in the technical feasibility of the tasks to be performed within the proposed schedules.

Although risk taking and risk analysis are part of the free enterprise system and influence most business decisions, the risks involved in the acquisition of software appear unusually high. This may be due, in part, to the large portion of money allocated to the software element of major weapon systems*. Many of the contractual issues concerning options and risks in acquiring software are described in Ref. [5]. There is a distinct difference between risks of a specific software development project as compared to a "high technological risk". On the one hand, a fairly simple software application may contain risks to either the Government, as purchaser, or the contractor, as developer. The risk, in this case, may be any identified area that potentially impacts the cost estimate in terms of increasing or decreasing those costs. Risks may be classed as cost risks (e.g., rate projections, overtime, technical

*The Worldwide Military Command and Control System (WWMCCS) is projected to spend over \$722 million on software, \$100 million on hardware; the Safeguard System spent an estimated \$467 million on software; and the Minuteman System spent \$124 million on Software (see Ref. [6]).

	FIXED-PRICE GREATEST RISK ON CONTRACTOR		COST-REIMBURSEMENT GREATEST RISK ON GOVERNMENT	
	FIRM FIXED-PRICE	FIXED-PRICE INCENTIVE	COST-PLUS INCENTIVE-FEE	
Applicability	<p>Fair and Reasonable price can be established at inception, e.g.,</p> <ul style="list-style-type: none"> - Reasonably definite design or performance specifications - Realistic estimates - Adequate competition - Valid cost or pricing data that provide reasonable price comparisons <p>Level of effort research contract</p>	<p>Where cost uncertainties exist and there is the possibility of cost reduction and/or performance improvements by given contractor (i) a degree of cost responsibility (ii) a positive profit incentive.</p> <p><u>Firm Target Type</u>: firm target and final profit adjustment formula can be negotiated initially</p> <p><u>Successive Target Type</u>: initial target can be negotiated, but firm final targets cannot; sufficient information will be available early enough in performance to set final goals.</p>	<p>← UNCERTAINTIES</p> <p>Development and test when incentive formula can provide positive incentive for effective management. Where feasible, use performance incentives together with cost and schedule incentives.</p>	(i) Level where measure evaluated under a definite measure
Essential Elements	Initial fixed-price places 100% responsibility and risk on contractor	<p><u>Firm Target</u>: target cost; target profit; price ceiling; and profit adjustment formula</p> <p><u>Successive Targets</u>: initial target cost and target profit; price ceiling; firm target profit formula; and production point for application to get either a firm target and final profit formula or a fixed-price contract.</p>	Target cost; target fee; minimum and maximum fee; fee adjustment formula (formula applied at end of performance)	Negotiated fee is evaluated
Limitations	<p>Government contractor must agree on fixed-price at inception</p> <p>If FFP level of effort, agreement must be reached on identification of effort and number of man-hours.</p>	<p>Adequate Contractor Accounting System Required</p> <p>Must determine (i) that this is least costly contract type and (ii) that any other contract type is impractical. Used for development and production procurements.</p>	<p>← Fee Limitations</p> <p>Production & Services R&D</p> <p>Formula should provide incentive effectiveness over variation in costs throughout the full range of reasonable foreseeable variation from target cost.</p>	ADDITIONAL (Same Final to dis of a develop have u
Approvals	None	Contracting Officer	← DETERMINATION AND FINDINGS	
Misc.	<p><u>Advertised or Negotiated Procurements</u></p> <p>Preferred over all types</p> <p>Minimum administration</p>	<p><u>Negotiated Procurements Only</u></p> <p>May also use performance or delivery incentives, where feasible.</p>	← NEGOTIATED PROCUREMENTS GOVERNMENT AUDITING A	Can use appropriate method

A

COST-REIMBURSEMENT GREATEST RISK ON GOVERNMENT		
EE	COST-PLUS AWARD FEE	COST-PLUS FIXED-FEE
UNCERTAINTIES IN PERFORMANCE -- IMPOSSIBLE TO ESTIMATE COSTS FIRMLY		
When incentive formula is incentive for effective feasible, use performance with cost and schedule	(i) Level of effort for performance of services where mission feasibility is established, but measurement of performance must be by subjective evaluation; (ii) work which would have been placed under another type of contract if performance objectives could be expressed in advance by definite milestones or targets susceptible of measuring actual performance.	Term Form: research, preliminary exploration, or study when level of effort is initially unknown (or development and test when a CPIF is impractical) Completion Form: research or other development effort when the task or job can be clearly defined, or definite goal or target expressed, and a specific end product required.
Fee; minimum and maximum formula (formula applied)	Negotiated estimate of cost: a base fee; maximum fee; the criteria against which performance is evaluated; resulting in an award fee.	Negotiated estimate of costs; fee fixed initially except for changes in the work or services required.
ADEQUATE CONTRACTOR ACCOUNTING SYSTEM REQUIRED		
ions ces ide incentive effective- n in costs throughout reasonable foreseeable get cost.	(Same fee limitations as CPFF and CPIF) Final fee determination by Government not subject to disputes clauses. CPAF is not for procurement of a major system categorized as either engineering development or operational system development which have undergone contract definition.	Fee Limitations - 10% estimated cost - 15% estimated cost Not for development of major weapons once explorations indicate engineering development is feasible.
DETERMINATION AND FINDINGS BY CONTRACTING OFFICER (EXCEPT FOR COST-SHARING)		
	(Same as CPFF and CPIF)	
NEGOTIATED PROCUREMENTS ONLY. "COSTS" DEFINED IN ASPR XV GOVERNMENT AUDITING AND ADMINISTRATIVE SURVEILLANCE		
	Can use combination of CPAF/IF or CPIF/AF as appropriate to reflect degree of subjectivity and objectivity of performance measurement method.	Least contractor responsibility for costs; least preferred contract type.

Figure 6. Contract Type vs Risk Considerations for Software

problems, technical performance, GFE interface problems, schedule/milestone slippages or constraints, special clauses, prime/subcontractor relationships, facilities, etc.) or fee/profit risks (e.g., fee/profit incentives, cost-sharing incentives, restrictions on costs, performance incentives, technical/schedule restriction on performance incentives, etc.). A high technological risk*, on the other hand, is the risk associated with a particular attempt to employ a computer or software in the solution to a problem that has heretofore not been solved via the computer. In other words, the problem/solution may be beyond the current state-of-the-art of computer technology.

In formulating the RFP, all high technological risks should be clearly identified as well as any non-obvious technical risks. All risks should be addressed in the Full-Scale Development Phase proposal by the prospective contractor via the Computer Program Development Plan (CPDP).

Formal Advertising	1-300.2 2-102
Negotiation	1-300.3 3-200
Types of Contracts	3-401 3-402 3-404 (Fixed Price Type) 3-404.2 (Firm Fixed Price) 3-404.4 (Fixed Price Incentive) 3-405 (Cost Reimbursement Type) 3-405.4 (Cost Plus Incentive Fee) 3-405.5 (Cost Plus Award Fee) 3-405.6 (Cost Plus Fixed Fee) 3-803
Special Types and Methods of Procurement	4-101

Figure 7. ASPR References Regarding Contract Type Selection

*Ref. [4] states that "The following software capabilities are likely to have technical risk;

- a. Certifiably correct control of access to data of different security classification and in different "need to know" categories;
- b. Automatic detection and correct reporting of equipment and software errors, and
- c. Automatic reconfiguration and recovery of the system from errors, including transition to and from degraded modes of operation."

Although certain types of contracts may limit the Government's liability, the contract type may also have the effect of obscuring the ability to track or collect total software development costs. Most Fixed Price contracts provide little actual cost visibility to the Government, except where progress payments are applicable. However, this is not the case when additional software development funds must be obtained via ECP. Although this type of additional funding requires a change to a specific functional/performance requirement, costs for each change must be tracked.

Ref. [5] discusses some issues related to procurement planning and types of contracts. It states the need for each procurement to be appraised to determine the issues (specific to the program) that impact contract type determination.

2.4 SUMMARY OF A SURVEY OF PROGRAM OFFICE SOFTWARE COST ESTIMATION PROCEDURES

The procedures for developing the system-cost estimate, particularly the software-related portion, consists of a logical progression of activities which update the original estimate as supporting data are acquired during the system's life cycle. This approach requires relevant historical data from analogous systems to form a basis for estimating initial costs. It further requires well informed software cost analysts and system designers to make engineering cost trade-off studies to determine cost differences between the system being acquired and analogous systems represented by historical data. While it is important for each software cost estimate to be realistic because it is a major component of the total allocated budget, the lack of available data on which to base comparisons with historical system cost data make it susceptible to gross error. The software cost estimate can seriously impact all decisions associated with a major weapon system. For that reason, it is important to derive a software cost estimate which is continually reevaluated and updated as additional information becomes available during development.

The software cost estimation process within the PO can be considerably improved with dedication to a systems approach which combines somewhat independent parts of the problem into an integrated methodology. Components of this methodology must necessarily include:

- Employing and educating qualified personnel with experience and knowledge of both the system acquisition and software development process. Understanding the complexities of acquiring and developing a major weapon system must necessarily draw upon information accumulated from both within the military organization and the software industry.

- Evolving a systematic cost estimating procedure which defines a series of steps providing for:
 - Definition of the objectives of the software cost estimation task.
 - Definition of personnel, resources, and time requirements needed for the software cost estimation process.
 - Description of the software requirements.
 - Identification of the software cost estimation techniques to be used.
 - Comparison and refinement of the Software cost estimates.
- Deriving guidelines on the sizing of software based on technical evaluation of the functional performance requirements of the system software.
- Collecting and analyzing software cost data from new and existing systems based on common definitions of data parameters to provide a historical cost element data base for derivation of cost estimating relationships and determination of factors which impact costs.
- Deriving and validating cost models supported by analysis of the historical cost data base.
- Ensuring that the procedures for software cost estimating are rigorously and methodically followed.

A survey of the software cost estimation process within the ESD environment was conducted recently (see Ref. [6]). Thirteen unidentified POs representing a diversity of ESD programs were contracted. Their responses are summarized by this author in Figure 8, and shows that a systematic approach for developing software cost estimates within the PO does not really exist. Many of the programs did not have any records to show that the software segment had been costed. Those programs that did derive a software cost estimate used varying techniques with inconsistent degrees of success. Although some programs were successful (see Program A in Figure 8), the methodology used does not appear to have impacted other programs' software cost estimating techniques. It is important for the level of technical detail concerning the software element to be accumulated and refined over the program life cycle. This includes documenting the assumptions made and cost estimating techniques used. In addition, the PO must employ knowledgeable and experienced personnel in the software cost estimation process. Lacking a standard and validated approach for software cost estimating requires that the PO use cost estimators who can draw upon analogous experience.

PROGRAM	IN-HOUSE ESTIMATION METHOD FOR SOFTWARE COSTS	COMMENTS
A	<ul style="list-style-type: none"> Decomposition of the software package to program modules, ranging in size from 600-85,000 instructions. Using a mean productivity rate obtained from a previous SDC study (325 JOVIAL instructions/month), total software cost was derived. 	<p>Estimated program size in instructions = 160,000; actual number to date = 156,000.</p> <p>Estimated cost 20% less than actual.</p>
B	<ul style="list-style-type: none"> A special team used analogy to existing system which new software/hardware was to replace. Parametric technique (e.g., Tecelote Model) based on number of instructions derived from the analogy. 	New system was to replace existing system.
C	<ul style="list-style-type: none"> Support contractor developed system design identifying 17 software modules by function. Estimate of effort in man-months for each module, resulting in a total man-month effort of 138-205 man-months. Additional 25% added to original estimate resulting in 173-256 man months or \$830,000 to \$1,270,000. PO increased original estimate to \$2.4 million due to uncertainties in cost estimates. PO again increased estimate to \$3.8 million to take into account program uncertainties. 	The estimate may have ensured sufficiency of budgeted funds, but the iterations resulted in a high estimate which may have also caused erroneous management decisions to be made elsewhere in the program.
D	<ul style="list-style-type: none"> No available data concerning the original cost estimate. 	Contractor's estimated software size was 20K instructions with a computer core size capacity of 132K instructions. Two years after contract award, software size was 174K, i.e., a 900% increase in number of instructions.
E	<ul style="list-style-type: none"> PO obtained software estimate size which was based upon a decomposition identifying functional modules from 200-50,000 instructions in size. PO multiplied the estimate by 1.5 to ensure that it was not optimistic. PO used \$150/instruction arriving at \$33 million. 	Due to inconsistent definition of terms, communication between the PO and the cost analyst led to an erroneous software cost estimate, which was fortunately discovered and corrected.

Figure 8. Survey of ESD Program Office Software Cost Estimation Methods*

*Summarized by author from Ref. [6].

PROGRAM	IN-HOUSE ESTIMATION METHOD FOR SOFTWARE COSTS	COMMENTS
F	<ul style="list-style-type: none"> No internal software estimate was prepared. 	<p>This program involved engineering changes involving software modification to an existing system.</p> <p>Analysis focused on the contractor's software estimate for each ECP.</p>
G	<ul style="list-style-type: none"> A joint software cost estimate was prepared by the PO and ESD Cost Analysis Division by decomposition. Productivity rate of 10 instructions/day/programmer was used. 	Although the productivity factor was obtained from unsubstantiated data, the total cost of software at project completion was 30% higher than estimated.
H	<ul style="list-style-type: none"> Estimate was derived from a decomposition process, but at a very gross level (i.e., the smallest module size was 12K instructions). 	Large module size may indicate a lack of knowledge of software requirements.
I	<ul style="list-style-type: none"> No data available. 	Difficult to learn from experience with a lack of historical data.
J	<ul style="list-style-type: none"> No data available. 	
K	<ul style="list-style-type: none"> MITRE and PO Engineers estimated number of instructions and an engineering estimate of resources/modules to yield cost estimate #1. The estimated number of instructions were multiplied by a cost per instruction factor of \$44 to yield cost estimate #2. By analogy, cost estimate #3 was derived. 	All three estimates were very close, providing the PO with a higher level of confidence in the estimate.
L	<ul style="list-style-type: none"> Sole source contractor estimated number of instructions which was then correlated with previous SDC productivity data for cost estimate #1. Using analogy and projection of software growth, cost estimate #2 was prepared. 	The estimates were close.
M	<ul style="list-style-type: none"> No data available. 	Contractor's cost proposal used decomposition and types of effort required to develop each module, using a productivity factor of three man-hours/instruction + \$15./instruction for documentation.

Figure 8. Survey of ESD Program Office Software Cost Estimation Methods (cont'd)

The conclusion reached following the survey and analysis of ESD cost estimating procedures follows:

"In 1976, DoD managers will be making decisions concerning the acquisition of an estimated \$3 billion of software...Based upon the research findings, there appear to be some major problem areas which inhibit the development of accurate and reliable software cost estimates..In estimating software cost, there are three possible sources of error. The first source of error is due to the element of change in the future which makes cost a random variable. A second source of error is the estimating technique itself...The third source is the non-uniform and unskilled application of a cost estimating technique. Regardless of how good a technique is, an accurate estimate may only be obtained if the technique is properly used" (see Ref. [6.]).

SECTION 3 - OFFEROR'S COST PROPOSAL PREPARATION

3.0 INTRODUCTION

This section presents information concerning the offeror activities associated with deriving and submitting a cost proposal in response to an RFP for a Full-Scale Development Phase contract.

Full-Scale Development Phase RFPs vary significantly in content* because of the differences in acquisition strategies, C³ system functional/performance requirements, program costs, schedules, and variations in the quality of Development (Part I) Specifications. This causes great variability in the amount of time and money allocated to the development of the technical and cost proposal for Full-Scale Development. When there is a Validation Phase, it is implemented most effectively by having contractors compete for the Full-Scale Development effort. The Validation Phase is the ideal time for contractors to develop a CPDP and SEMP. That way, they each evolve a design approach supported by trade studies, management plans, and support plans. In the event there is no Validation Phase, the amount of manpower allocated to developing an optimum top-level design approach in response to an RFP depends to a great extent on the respective financial positions of the competing offerors. The effect of the different acquisition strategies and the financial positions of the offerors are demonstrated by the level of technical detail and supporting studies in the proposed system design. Estimated costs for system development are more realistic when they are based on a thorough understanding of the technical problem to be solved, and the proposed solution has been derived by examination of alternative system designs in terms of cost, schedule, and performance requirements. For that reason, the system acquisition strategy should take into consideration the requirements necessary for producing a realistic cost proposal in response to an RFP.

*For example, (RFP(s) for Full-Scale Development should include an authenticated System/Segment Specification developed during the Conceptual and Validation Phases as well as the authenticated Development (Part I) Specification. The contract specification will be the authenticated Part I's providing they have been approved and authenticated at the end of the Validation Phase. If the Validation Phase was conducted in-house, the CPCI Development Specification, developed by Air Force personnel, will accompany the RFP. If the Validation Phase had competitive contractors evolving a design approach, the Development Specification that was developed during the Validation Phase is placed on contract. If there was no Validation Phase at all, no Development Specification accompanies the RFP as it will be developed and produced by the winning contractor early in the Full-Scale Development Phase.

This section is applicable to the offeror's cost estimation process regardless of the specific acquisition strategy. Although a competitive Validation Phase allows for more extensive technical analyses and cost-performance trade-offs, the cost estimation process is essentially identical once the design approach has been formulated. In 3.1, various cost estimating techniques used by industry are discussed. In 3.2, an analysis is presented of the many issues which impact software development and complicate the estimator's ability to accurately predict costs and schedules. In 3.3, the offeror's methods for representing the scheduling of the work packages is discussed.

3.1 COST ESTIMATING TECHNIQUES

A number of cost estimating techniques are used within the software industry. They are often referred to by different names, are sometimes used in combination, or are slightly different in purpose or application. This discussion briefly describes the major techniques currently used.

The purpose of an offeror's cost proposal is to present his estimate of the costs (along with a reasonable profit) to develop a technical solution to a given set of performance requirements within a schedule acceptable to the Government. To prepare an accurate and complete cost estimate, a contractor must perform the following tasks:

- Develop a technical solution (i.e., a top-level system design) which is responsive to the SOW/RFP functional/performance requirements, including a logical and consistent definition of the tasks, work packages, and products necessary to produce the operational software representing that system design.
- Apply costing algorithms or criteria to estimate the costs associated with each of the elements derived above.

Generally, the project definition process decomposes the total software development task into units small enough to be more accurately examined for purposes of subsequent cost analyses. Decomposition of a total development project into its constituent parts is almost always a prerequisite to other cost estimation techniques*. (This does not necessarily imply that the top-level system design evolved by the proposal team becomes the contractual baselined design. Rather, this process is a validation of the offeror's requirements analysis.) It may be useful to decompose a system into software elements (i.e., programs and subroutines) or work units (i.e., design, code, and test activities). A possible result of this decomposition is a more accurate cost estimate. An estimate of total development based on an aggregate

*Charles Lecht (see Ref. [7]) asserts that the analysis of system requirements must identify not only the requirements to be satisfied and the activities to be performed, but also a means of recognizing completion of activities, thereby requiring a comprehensive decomposition of a software system and extensive project planning.

of smaller estimates, where each is made on a constituent part, may have a smaller error because the error in each is smaller. If the cost estimates for the constituent parts are made independently and as precise as possible, then it is statistically possible for random errors to cancel out each other. However, if each estimate contains padding, the aggregate estimate will be too large.

3.1.1 Analogy of Similar Experience

The analogous cost estimation technique bases the cost of the proposed software system, or portions thereof, on costs actually required to produce one or more similar software segments. Adjustments to the derived cost estimate are made for any differences that are found between the new system and existing ones. For the analogy costing technique to be effective, the software function/performance requirements must be quite similar. In addition hardware characteristics of the existing and proposed systems must be carefully evaluated to determine the relationship of the software systems. The total elapsed time for development of both software systems must also be examined.

The analogy method of cost estimating is one of the most widely used techniques in the industry. Since this estimating method depends upon data obtained from similar projects, it is valid only when supported by a cost element data base that contains cost data as well as a selective set of each development project's hardware/software/schedule characteristics. If software contractors do maintain such a data base, it generally contains sensitive information which is open to scrutiny only by selected corporate personnel, and not necessarily the people preparing the cost estimate. However, few comprehensive cost element data bases of this nature actually exist within the industry. In lieu of a cost data base, this method relies upon the cost estimator's skill in making analogies as well as his recall ability. While knowledge of similar software application areas and familiarity with the responding organization's capabilities and performance on previous projects is a valuable asset, reliance on the ability to recall functional/performance/cost characteristics of systems over time is a poor substitute for a cost data base.

The analogy method has been criticized for both the lack of a valid historical data base of performance, cost, and schedule data, and the non-linear relationship between system cost and system software size which perturbs analogous comparisons. However, the analogy approach has proven to be fairly accurate if the development projects are relatively small and similar in operational characteristics, and if adjustments are made for growth and improvements in technology and management of software development.

3.1.2 Quantitative Method

This cost estimation technique divides the total software development effort (including support and other non-deliverable software) into work packages or units, which may be performed by a single individual. Once the total effort

is subdivided, the number of work units is multiplied by a previously determined cost-per-unit factor or productivity factor, derived from estimates of software complexity and project duration. Often the quantitative method is iterated as progressive levels of detail on the system are available. Software development factors unique to the project are often evaluated, reduced to a single weighting factor, and used to modify the derived estimate. This method relies heavily upon the ability to estimate total number of instructions and programmer productivity.

This type of cost estimating method has been espoused by Wolverton, Aron, Meyer, and Weinwurm (see Refs. [8, 9, 10, & 11]). A basic disadvantage of the many versions of this technique is the subjective assessment of the weighting factor used to modify the derived estimate. Also problematic is the previously determined cost-per-unit factor because it is not always clear what that cost includes (i.e., direct labor, direct labor plus overhead) and the unit (i.e., machine instruction, source statement) is often incomparable between development projects. This cost estimation method also needs to be supported by a valid cost data base consisting of comparable and consistent project data from which factors, such as productivity,* may be realistically determined. This method also relies heavily upon the individual estimator's experience and ability to evaluate each software development project in terms of the internal contractor environment in which it will be performed.

3.1.3 Percent-of-Other-Item Method

This cost estimation method determines the net development time in terms of man-days. This figure is obtained by calculating the development time necessary for the detailed design of CPCs, code of CPCs, generation of test data, test of CPCs, and document preparation. Analysis and design of the CPCI are omitted from the estimating algorithm which leads one to believe that this method of cost estimating is unsuitable for C³ systems. The complexity of the functional/performance requirements of weapons software demands that resources be allocated for CPCI analysis and design activities and that the schedules devised for CPCI analysis include realistic time budgets. Once the net development time is calculated, other project-dependent characteristics

*Software productivity figures are extremely sensitive to definitions so they are often non-comparable. Programmer productivity is generally a ratio of the number of deliverable instructions (source or object) produced by the average programmer per unit of time. The number of non-deliverable instructions are also items affecting cost. Complexity of the software and choice of programming language may be important factors in deriving a productivity ratio. Productivity can be defined in quantitative terms from measures in a data base if software development data is kept current and accurate. In addition, data parameters reflecting the software methodologies used in the various phases of the development process must be recorded in order to ascertain their effect on the productivity ratio.

(e.g., program complexity, programmer know-how, and programmer job knowledge) are analyzed and weighted to establish the net program development time estimate. The net development time estimate is then further modified to account for the factors of other system time (time required for the design, test, installation, and maintenance of the software system), project-loss time (time which is charged to the project, but which is non-productive or indirectly productive), and non-project time (time which is spent on activities not related to the project).

This method has been criticized for both possible and probable error magnification since the net development time is the base for all subsequent calculations and may contain gross errors. The skill required on the part of the estimator for assigning weighting factors to project-dependent variables (e.g., programmer know-how, programmer job knowledge) may also be a source of error besides obscuring comparability of cost data because of subjective bias.

3.1.4 Rules of Thumb

Numerous software development guidelines, or rules of thumb, have come into existence in the process of estimating development costs. Rules of thumb may be examples of individual or collective experience and may be close approximations of actual resource/schedule requirements. Rule of thumb estimates are often used in conjunction with other estimating methods. Figure 9 identifies typical rules of thumb.

3.1.5 Parametric Equations

A parametric equation is a mathematical representation, or function, used to project the cost of a proposed system by using variables, or parameters, which have been analyzed by previous software development experiences and for which a known, or quantified, relationship exists. The types of variables used in parametric equations are generally cost items, such as technical manpower, support and management manpower, and computer resources. Statistical methods generally used in parametric equations include scatterplot analysis, correlation analysis, analysis of variance/covariance, multiple regression analysis, and factor analysis. Parametric equations can be used for comparative analyses, but the unreliability of the original cost parameter values makes the absolute costs essentially useless.

DEVELOPMENT VARIABLE IMPACTING COSTS	SOURCE	RULE OF THUMB
<ul style="list-style-type: none"> Allocation of Time to Full Scale Development Activities 	Aron (see Ref. [9])	Planning - 30% Coding - 20% Testing - 50%
	Brooks (see Ref. [17])	Planning - 33% Coding - 17% Testing - 50%
	Nelson (see Ref. [34])	Planning - 34.5% Coding - 18% Testing - 47.5%
	Wolverton (see Ref. [8])	Analysis/Design - 40% Code/Debug - 18% Test - 40%
	Morin (see Ref. [33]) or Farr (see Ref. [12])	Design Total System - 1 to 3 Man-Months (dependent on conditions and delays experienced.) Design Computer Program System-10% of Total Man-Months Design Programs-1 Man-Month per 1000-2000 Machine Instructions Code Programs-1 Man-Month per 5000 Machine Instructions Test Individual Programs - 20% of Test Effort Test Subsystem-Approximately 50% of Total Test Effort
<ul style="list-style-type: none"> Computer Resource Requirements in Elapsed Time for FSD Activities 	Aron & Arthur (see Ref. [35])	Six Hours/Programmer/Month (For Programmers with a Good Understanding of the Data Processing Application) Eight Hours/Programmer/Month (for Unfamiliar Applications) Twelve Hours/Programmer/Month (For Real-Time Projects)
	Meyers (see Ref. [10])	Three Hours/Month x Number Development Programmers Through System Design Phase Twelve Hours/Month x Number Development Programmers Through Implementation Phase Fifteen-twenty Hours/Month x Number Development Programmers Through Integration and Test Phases

Figure 9. Rules of Thumb

DEVELOPMENT VARIABLE IMPACTING COSTS	SOURCE	RULES OF THUMB
<ul style="list-style-type: none"> Complexity of Software 	<p>Graver, et al (see Ref. [22])</p> <p>Doty (see Ref. [1])</p>	<p>Four Hours/Man-Month Twenty Hours/1000 Instructions</p> <p>Real Time Applications Are More Complex Than Non-Real Time Applications</p> <p>Scientific Applications Are More Complex Than Business Applications</p> <p>Operating Systems Are More Complex Than Compilers Which Are More Complex Than Applications Software</p> <p>Support Software Is More Complex Than Applications Software</p>
<ul style="list-style-type: none"> Documentation 	<p>Aron (see Ref. [9])</p> <p>Doty (see Ref. [1])</p> <p>Morin (see Ref. [32])</p> <p>Geran (see Ref. [33])</p>	<p>Easy - Very Few Interactions With Other System Elements</p> <p>Medium - Some Interactions With Other System Elements</p> <p>Hard - Many Interactions With Other System Elements</p> <p>Thirty Pages per 1000 Lines of Source Code Five Pages per Man-Month</p> <p>Ten to Thirty-Five Pages of Documentation/100 Lines Program Code</p> <p>Two Man-Months/User's Document Three to Five Pages/Man-Day for Draft, Using 750-1250 Words/Page</p> <p>Twenty Pages/Man-Day for Technical Review Fifty Pages/Man-Day for Editing Ten to Fifteen Pages/Man-Day for Typing</p> <p>Ten Percent (approximate) of Total Development Cost Thirty-Five to One Hundred Fifty Dollars/Non-Automated Page</p>

Figure 9. Rules of Thumb (cont'd)

3.2 ISSUES IMPACTING SOFTWARE DEVELOPMENT COSTS

Commencing with studies performed by System Development Corporation and Planning Research Corporation (see Refs. [11 through 15]) in the early 1960s, the factors that impact software development costs and schedules have been examined on many occasions. Several factors have been identified by research groups or collective opinion (e.g., the October 1974 Electronic Systems Division/Government/Industry Software Costing and Sizing Workshop). The result of all this analysis is a large store of documentation, little consensus, and a disappointingly small advance in the state-of-the-art of software cost estimation. Indirect benefits have resulted from the widespread recognition of the need to determine what effect (or relationship) the various software development variables have on the cost of acquiring and supporting a system. To develop improved software cost estimating relationships, it is recognized that a cost data base containing a collection of consistent and comparable data must be available for statistical analysis.*

The following paragraphs present a brief condensation of some of the major factors thought to impact the software development process. A brief overview discussion of these factors can be found in Appendix III of Ref. [16]. In addition, a comprehensive and recent discussion of these factors is presented in Ref. [1].

3.2.1 Complexity of Application

The complexity of software under development is one of the more important factors impacting the development process in general, and costs and schedules in particular. Complexity effects programmer productivity as measured in output per unit of time. Programmer productivity varies with the type of development job, and therefore, accuracy of productivity estimates is questionable. The exact relationship between programmer productivity and complexity of the application is unknown because of the creative nature of the task, the external attributes of each software problem, unique individual differences, and the variability of terms in measuring output per unit of time. Consequently, the exact relationship of programmer productivity and program complexity is unknown, and will probably remain a subjective assessment for some time.

*Rome Air Development Center (RADC) is currently in the process of establishing a data repository of software development parameters consistently collected from a wide range of development projects. The purpose of the RADC repository is to gather user experience for the study of software development (specifically, software reliability, programmer productivity, and software development costs) by members of Government, industry, and academia.

The complexity of a particular software application affects many other aspects of the development process (e.g., design and testing). For that reason, it is often quantified by subjective assessment or by rule of thumb and included in costing algorithms. (See Figure 9 for rules of thumb concerning complexity of software applications). The derivation of a software complexity index or assessment involves the determination of the application's characteristics and may be the most important reason for identifying the particular type of software application, but even within a particular application there are numerous other factors (e.g., quality of functional/performance specifications) which may complicate quantifying the complexity of the application.

A complex application that may involve innovative or high-risk technology should be so identified in the RFP if possible.* In any case the offerors should identify any areas which they detect as high risk in their proposal. The offeror's technical proposal should demonstrate his appreciation of the complexity of the problem by his estimate of required man-months and his allocation of time to complex development activities. A brief discussion of high risk technological areas may be found in Appendix III of Ref. [16].

Doty, Ref. [1], suggests the following guidelines for resource allocation:

- In estimating productivity rates for avionics applications, the cost estimator should assign separate (and ascending) productivity rates for development of on-board flight programs, simulation, and automatic test equipment, in that order.
- Command and control developments should be considered to have an approximate 40 percent decrease in productivity rates due to their real-time requirements, their large size (an average of 500,000 object words) and complexity of control flow relative to software with simple flow of control.
- Business application developments show a higher productivity rate than non-business applications. Alternative size or cost estimating algorithms may be appropriately based on the number of input/output items or the number of processing transactions for business application estimates.
- Scientific application development should use a lower productivity rate for cost estimation because of their use of complex computational algorithms, although the productivity rate in these developments is highly dependent on other factors such as use of a Higher Order Language (HOL), real-time requirements, and Central Processing Unit (CPU) time and memory constraints.

*An example of high risk technology is formal program verification.

Brooks, Ref. [17], presents data which suggests that productivity rates may be impacted more by the choice of a HOL than by the complexity of the application, (e.g., the use of a suitable HOL will increase productivity rates no matter how complex the applications).

3.2.2 Total Software Size

Another important component of any software development project is the estimated size of the software. Doty, Ref. [1], states, "Estimating the size of software programs has proven to be the most difficult aspect of, and the source of greatest error in, analyses to project resource requirements of software development. The size parameter is used in nearly all cost estimating models and numerous studies have been performed in an attempt to identify the relationship between software size, costs, and schedules. However, in spite of its general use, there are numerous problems associated with software size effects. Basically, the problems are as follows:

- Software size estimates are generally given in number of object instructions; however, some estimators define instructions as being source statements. Proponents of source statement usage argue that since source statements represent the programmer's output, productivity rates should reflect programmer output per unit-of-time. Others insist that object code, being the output of the compiler, measures programming output more accurately. Because the programmer constantly changes and corrects his source program, he produces more source code than appears in the final product. Therefore, it is easier to measure the final number of object code instructions in the software than the programmer's total output. One may use either object or source statement measurements, but not both, and the distinction must be clearly made. Because source statements offer a more stable statistical unit, size estimates should be based on number of source statements.
- The use of a HOL source statement productivity rate does not generally differentiate the HOL used. Not all HOLs can be used with the same facility for all applications.
- The expansion ratio derived for HOL may vary between HOL, compilers for the same HOL, or different operating systems. This consideration is important when sizing estimates are based upon analogous and existing software.
- The software cost estimate must be based in part on the total number of instructions needed for delivery of the contracted product. The estimated number of instructions used may include software that must be developed but not delivered. This is especially true in weapons systems where a large amount of

software is needed to support the development of application software. Such software may consist of compilers, simulators, utility tools, test tools, library systems, and other support programs. Comparisons with analogous development projects clearly must account for the possible inclusion, or exclusion, of both deliverable and non-deliverable support software in the costing process.

- Although it is commonly accepted that the size of the software (whether measured in lines of code, number of deliverable products, or number of program units) is related linearly to cost, a standard productivity rate used for small development efforts has little relationship to the productivity rate derived for large systems where intercommunication and coordination are not measurable attributes of a productive day. The manpower required to build a complex C³ system is enormous, requiring tasks to be divided into sub-tasks and sub-subtasks. The effort required for intercommunication and coordination between all tasks increases the time required and subsequently costs. Brooks, [Ref. 17], states, "If each part of the task must be separately coordinated with each other part, the effort increases as $n(n-1)/2$." In this instance n is the number of separate tasks.

"Since software construction is inherently a systems effort (an exercise in complex interrelationships), the communication effort is great and it quickly dominates the decrease in individual task time brought about by partitioning. Adding more men then lengthens, not shortens the schedule." (See Ref. [17].)

3.2.3 Requirements Specification

The Computer Program Development (Part I) Specification for a CPCI is derived from the System Specification. This derivation may be accomplished by Validation Phase contractors or in-house personnel, depending upon the acquisition strategy. The types of requirements defined must be examined for both feasibility and impact on the allocation of resources to the development project. In addition, the completeness, complexity, rigor, and compatibility of the CPCI performance requirements must be appraised.

According to Ref. [1], the following types of performance requirements affect productivity in respect to given standards and, therefore, time and costs:

- Special display programming for display equipment and plotters may result in a 10-30 percent decrease in productivity.
- Real-time operation where response times may be critical may result in a 25-70 percent decrease in productivity.

- CPU time constraints may reduce productivity by 25-57 percent.
- Memory size constraints may reduce productivity by 15-30 percent.
- Concurrent development of the hardware components required to interface with the CPU in the operational environment may result in a 20-55 percent decrease in productivity.

It is recognized that the quality of the functional/performance requirements has an enormous impact on the development process, especially costs and schedules. Too little detail allows for ambiguities in interpretation. According to Ref. [1], the effect of vague operational requirements on productivity and, therefore, time and costs, is as follows:

- Command & Control - 25 percent decrease.
- Scientific - 50 percent decrease.
- Utility/Business - No effect because operational requirements for these programs are usually adequately defined before design.

Highly detailed functional/performance requirements usually result in a specification of design implementation for a particular performance requirement. The eventual effect of specifying the design in the Development Specification is to shift greater responsibility for the design of the end product (and perhaps cost) to the Government. If the contractor implements a design according to design specifications that are later found not to satisfy a functional/performance requirement, the Government must eventually pay for redesign through ECPs, since the original design was so specified by them.

No widely used and acceptable requirements analysis language for requirements specification currently exists although several requirements and design languages are emerging in the industry, including the Computer Aided Design and Specification Analysis Tool (CADSAT formerly CARA), the Software Requirements Engineering Methodology (SREM), and the Specification and Assertion Language (SPECIAL). When these tools become more reliable and available, the requirements specification process may have less of an impact on development costs and schedules.

3.2.4 Level of Change in Performance Requirements

During software development, the contractor must expect some change in performance requirements. Depending upon the volume, extent, and frequency of the changes, the software development process may be severely interrupted or altered. Each change processed by the contractor will have the effect of making some aspect of the partially completed product obsolete. If the Development Specifications are of poor quality (i.e., incomplete, inconsistent, or vague), the software cost estimator must realize that changes in requirements will result, although the impact of change on the development project is generally not an issue addressed by the cost estimator. Because contract types differ (i.e., Fixed Price or Cost Reimbursement), the contractor must consider the provisions necessary for processing changes, costs associated with change provisions, and the specific type of development contract. The impact of change on development costs and schedules must be evaluated for each change processed.

3.2.5 Documentation Requirements

Documentation requirements for a system acquisition are very costly* and for that reason the Air Force Acquires only that documentation which is specifically required. However, some software acquisitions require an inordinate amount of documentation. Just as one analyzes the components which determine the resources necessary for the development of software (i.e., complexity, size, schedule duration and adequacy, stability of requirements, and multi-contractor interfaces), so must the cost estimator analyze the cost factors impacting the preparation and acceptance of required documentation.

The estimator must consider the documentation requirements of the specific design approach outlined in the contractor's technical proposal. Further, he must be aware that when a CPCI is specifically designated by ESD for procurement, it may result in an unrealistic proliferation of configuration management, program control, and technical progress documentation. Section 2 of Ref. [18] discusses the problem of numbers of CPCIs, as does Section V of Ref. [19].

*The Government/Industry Software Sizing and Costing Workshop states that documentation costs approximate 10 percent of the total software development cost, or \$35-\$150 per page, depending upon the amount and complexity of the analysis required in document production.

Delivery and acceptance of documentation depends to some extent on its timeliness. In estimating schedules and resources for tasks, the cost estimator must consider the delivery dates of documents in relation to other milestones, availability of information requirements (i.e., supporting documentation data obtained from decision points), and the resources required for documents preparation.

3.2.6 Software Quality Requirements*

Software quality attributes have undergone extensive examination by both Government and industry. Ref. [36] discusses software quality factors, their definitions, associated relationships between the factors, and uses for identifying the trade-offs between the conflicting quality factors in determining the product's required capabilities and performance characteristics. Quality attributes such as reliability and maintainability, are currently being specified by the military in terms of performance requirements. However, these types of quality requirements are not yet absolutely quantifiable in the current state of software technology, and their specification in a performance requirement generally results in a subjective interpretation rather than an objective measurement upon which acceptance of the product depends. Another problem of specifying quality requirements in a software procurement is that many of the software quality characteristics are in conflict with each other, e.g., modularity and efficiency. Because it is difficult to reconcile such conflicting requirements, the software engineer may be forced to make arbitrary, but less than optimum, design decisions. Yet another problem is the lack of standard criteria for quality metrics, such as testability or portability. The overall effect of the imposition of quality requirements on the development process appears to be increased costs for development activities, but decreased costs for maintenance and support activities. (See Ref. [1]).

3.2.7 Software Development Schedule

The offeror's cost estimate is based upon a specific project definition, a plan and schedule for task performance, a set of mixed personnel skills, necessary assumptions about many of the factors known to impact the development process, and his individual costing rules. The entire cost estimation process must be iterative because all of the variables (especially schedules) and the relationships between those variables are unknown at the onset of the process. In addition, there is a practical need for the allocation of resources to be appropriately spread over the entire development cycle and be so specified in the cost proposal to facilitate preparation of a time-phased budget.

*For more information on software quality requirements, see MIL-S-52779(AD) and Ref. [37].

The total amount of calendar time allocated for software development has a significant impact on costs. Much analysis has been performed on the relationship between total development time and total development costs. The cost estimate for software development must allocate resources over the total elapsed time to:

- Plan for time-phased funding.
- Allocate resources for all explicit, derived, and implied tasks resulting from the breakdown of requirements.
- Account for costs that are a function of time and vary with the time they are incurred (i.e., computer utilization).
- Manage the project within budgeted resources and schedules.

Generally, the development schedule is a fixed constant. A critical component of the derivation of the cost estimate is how the contractor specifies the initiation and termination of identified tasks within the time constraints. Because most development tasks are sequential in nature, they cannot arbitrarily be compressed or reorganized within the allocated schedule. In other words, the number and sequence of tasks to be performed in a given time period will indicate the manpower required to perform the total job.

There is little non-proprietary, quantitative data available for effectively measuring the cost impact of a less than optimum development schedule. There is data available, however, depicting the relationship of program size to development time. For instance, there appears to be an optimum man-loading algorithm, loading above or below which will negatively impact costs and schedules. Figure 10 depicts the distribution of manpower for a medium size software development project. Further, the following management implication must be realized by cost estimators:

"Management cannot diminish the development time of a system without increasing the difficulty. All changes take place in the negative time direction. Development time is the most sensitive parameter. It cannot be set arbitrarily by management." (See Ref. [20].)

The manner in which the contractor allocates the time to specific activities and individual skills is also of prime importance. Too little time and effort spent in analysis and design has an enormous impact on the eventual costs to subsequently correct design deficiencies. As development progresses, it becomes more and more costly to resolve design errors.

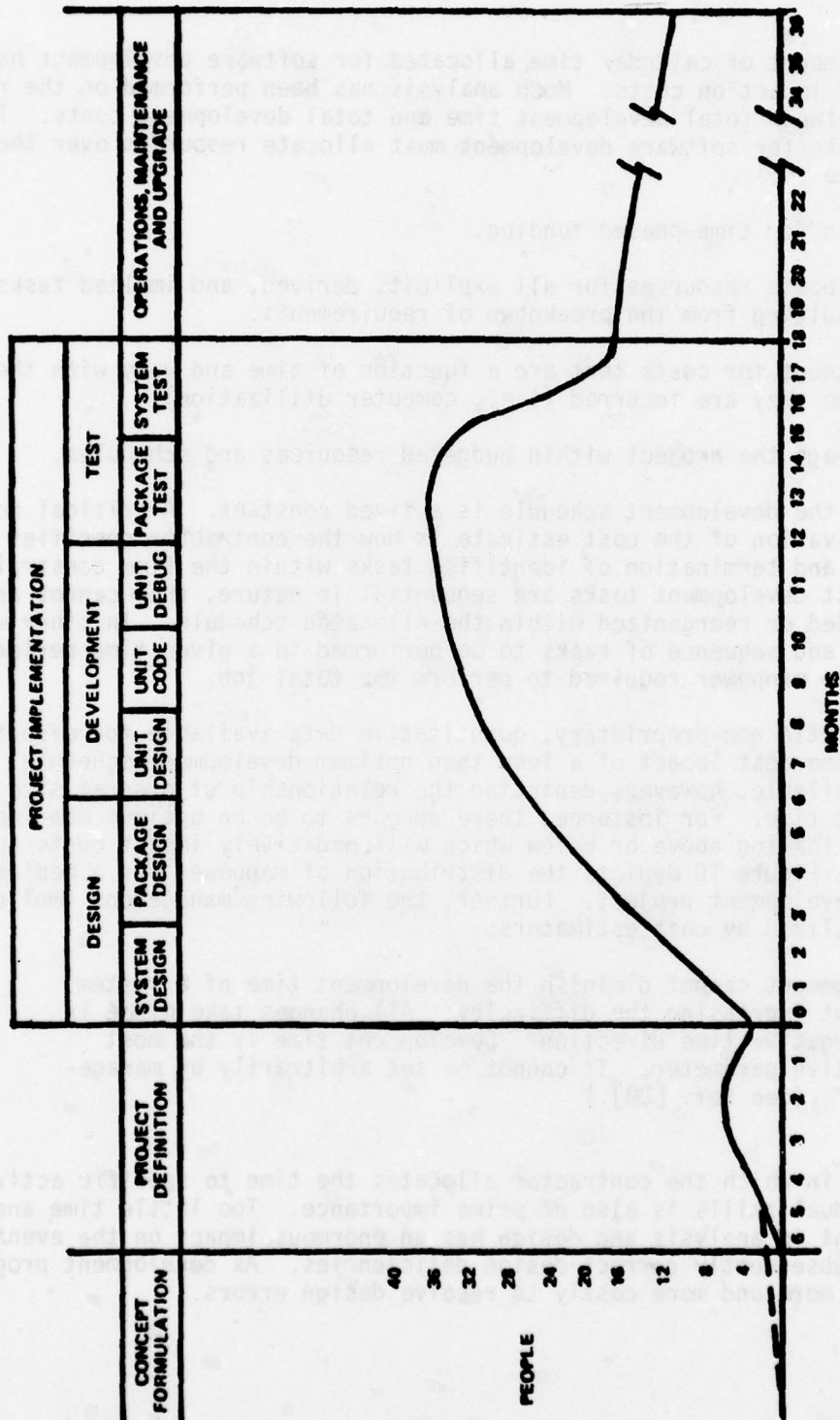


Figure 10. Estimated distribution of resources for a Medium-Large Project. (See Ref. [1].)

There is an optimum man-loading as a function of time in the development process, and deviations from the curve by the addition of more manpower may have an inverse effect on completion time. This is due (1) to the natural division of the work packaging, below which it is unprofitable to sub-divide, (2) to the amount of training required of personnel assigned to do the work, and (3) to the amount of human interfacing and intercommunication necessary to produce a given product.

"Adding manpower to a late software project makes it later... The number of months of a project depends upon its sequential constraints. The maximum number of men depends upon the number of independent subtasks. From these two quantities one can derive schedules using few men and more months. (The only risk is product obsolescence.) One cannot, however, get workable schedules using more men and fewer months. More software projects have gone awry for lack of calendar time than for all other causes combined." (See Ref. [17].)

The costs added to a late project by adding manpower may be more than those incurred by the additional manpower cost. There may also be further costs resulting from the additional training and coordination required. The increased complexity in the development process caused by additional manpower can cause the project to fall further behind the schedule.

3.2.8 Type of Software Development Effort

Software development cost estimates vary according to the amount of new code to be generated, transferred, or retrofitted.

Transferring an operational software system to a new equipment configuration is a cost estimation problem in which some of the variables need to be analyzed in a unique manner. (See Appendix B for a discussion of the Hahn and Stone software transfer model.) Costing individual changes or retrofits to an existing software system presents other problems, especially if the developer is not familiar with the existing system. The requirement for high quality documentation accompanying the existing software is obviously more stringent for retrofit development efforts. Costing software retrofits must include analysis of the existing system, decomposition of the retrofit requirements, and estimation of the costs of modifying the existing programs to interface with new software. At some point, the contractor may find that it costs more to retrofit the system than to rewrite it.

3.2.9 Personnel Requirements

Software development is a highly analytical, and sometimes creative, endeavor requiring individual and collective abstract reasoning to deal with complex problems. The issue of manpower in cost estimation analysis is generally

reduced to deriving a productivity figure per manpower unit for an average person within a skill category. Due to the individual skill requirements of each software development effort, a standard skill distribution ratio is not possible to derive. However, the following rules of thumb have been suggested (see Ref. [1]):

- The distribution of support personnel (i.e., management, clerical) to system programmers and analysts is 20 percent support to 80 percent programmers/analysts.
- The effect of this skill distribution is to increase costs (i.e., per unit line of code) 25 percent.

Early System Development Corporation studies performed by Sackman, Erickson, and Grant (see Ref. [23]), reflect a variation in productivity rates for experienced programmers of 10:1. Further, there was little correlation between performance and experience. Many software contractors use internally-generated productivity ratios for estimating software development costs, but because of the numerous and unique combinations of factors inherent in each development effort, the productivity ratios may be no more than an average guideline.

Perhaps the use of an application-suitable HOL is the factor most likely to impact collective productivity averages. Programmer productivity has been seen to increase by as much as a factor of five with the use of a HOL (see Ref. [17]).

3.2.10 Development Methodology

Although the development approach for software is generally the contractor's responsibility there is a trend in current procurements for the implementing command to specify the use of development methodologies to provide increased visibility and control. A specific methodology is generally required to ensure that the contractor is responsive to software quality factors, such as maintainability. Certain development techniques (i.e., structured programming) are perceived to increase productivity and lower maintenance costs. Although there is little hard evidence that the use of specific software engineering methodologies on large C³ systems will result in lower development and maintenance costs, the consensus of informed developers supports the premise. There are, however, other factors that must be considered by the contractor in responding to the RFP that specifies how the software is to be developed, including:

- In what language is the software to be written? Although the programming language selected can be considered part of the development methodology, it may be chosen because of factors external to that methodology. It has been observed that the use of a HOL increases programmer productivity and, therefore, use of a HOL should decrease development costs. However, some applications are constrained by core space or time critical parameters, requiring that the software be written in a Machine-Oriented Language (MOL). The use of a HOL is preferable and should result in lower development and maintenance costs.
- How is the software to be verified? When verification requirements include the use of an Independent Verification and Validation (IV&V)* contractor or use of testing tools that have the effect of verifying that a certain percentage of the software system statements have been executed within a given set of test cases, the contractor must expect increased development costs.

3.3 PROJECT MANAGEMENT AND SCHEDULING PLANS

A prime objective of the offeror's proposal preparation process is to establish a framework of plans to demonstrate his methodology for monitoring development costs and schedules. Plans are necessary for tracking project schedules and costs, and for coordinating all activities necessary for delivery of the product. Although the project schedules are preliminary in nature during proposal preparation, the offeror must convince his own management, as well as the procuring agency, that sufficient project planning and analyses have been performed to verify that the product as specified can be built within cost and schedule constraints. The project plan contained in the Computer Program Development Plan (CPDP), should demonstrate the feasibility of developing the software within time and costs.

*An IV&V requirement may be specified in two ways: (1) special tasking by a separate organization within the contractor's shop; (2) an independent agency or contractor different from the developer. The IV&V role is that of an observer or testor whose purpose is to remove possible bias in contractor test analysis. When an IV&V contractor is used in a procurement, provisions for his free access to required information, documentation, test tools, test facility, and source code listings must be assured. This requires that contractual provisions be well defined so that each contractor's responsibilities are fully compatible.

Numerous methods exist for representing activities and their duration, although generally, fairly simple schedule charts are included in the proposal. Bar charts, which are used in production management, have been replaced by more sophisticated scheduling representations that show complex interrelationships of project tasks, subtasks, and products within a given time frame. Network planning methods may be the optimum mechanism because they graphically represent the precedence relationships, or dependency of activities, in a more disciplined manner. They can be used to determine the effect of schedule changes of activities on those occurring later in the project. Figure 11 depicts examples of the Gantt, Project Network, and Time-Scaled Network charts.

In a large development project consisting of perhaps hundreds of activities, it is necessary to reduce the project plan to a series of interconnecting activities and events which result in a network model of the plan. Often the magnitude of the project requires the contractor to organize the networks by various subsystems or other divisions, corresponding to the different levels of project management. The WBS provides an effective level of detail for project network charts if it represents activities to the CPC-level. Since the WBS is used in cost accounting and reporting, it is important for the contractor to develop and represent the project schedule with WBS uniformity.

Generally, a complex network of events and critical path identification is not submitted with the proposal. The critical path in a network chart is defined as the sequence of activities which consumes the most estimated time in reaching the project's end event. Slippage of activities along the critical path will cause the end event to slip; thus activities on this path require the most management attention to avoid slippages. Because critical path analysis is a time consuming task in a large development project, this type of analysis is generally derived from the initial project schedules after contract award. This practice, however, may obscure important schedule constraints that should be evaluated for their impact on development costs.

As noted, initial schedules and milestones are submitted for evaluation in the CPDP when it is prepared by the contractor for inclusion in the proposal to be used in source selection. In this case, the CPDP is identified in the RFP as a required proposal document. There may be some overlap of information (e.g., schedules for milestones, identification of deliverables) in the CPDP with information presented elsewhere in the proposal or eventual contract. (See Appendix C of Ref. [4]). It is important for this information to be consistent as well as correct.

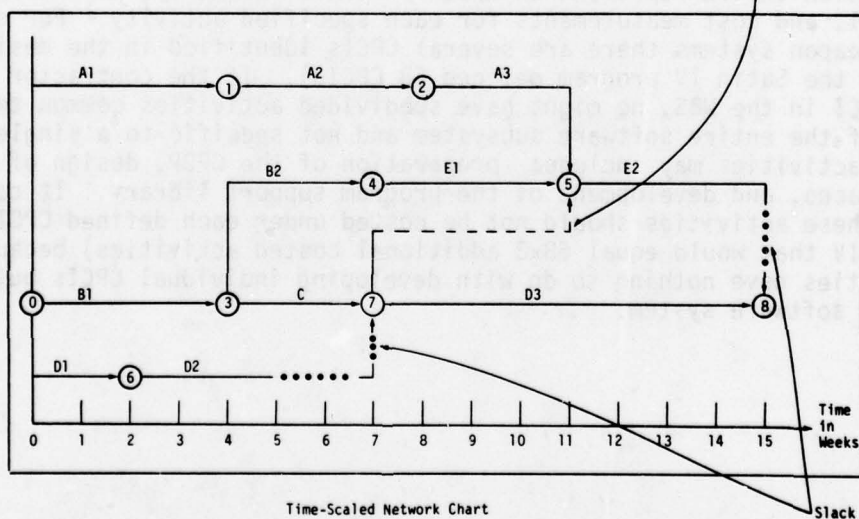
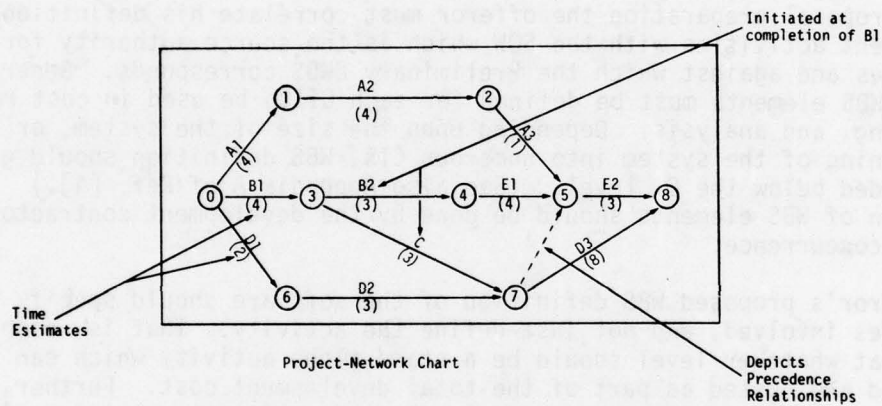
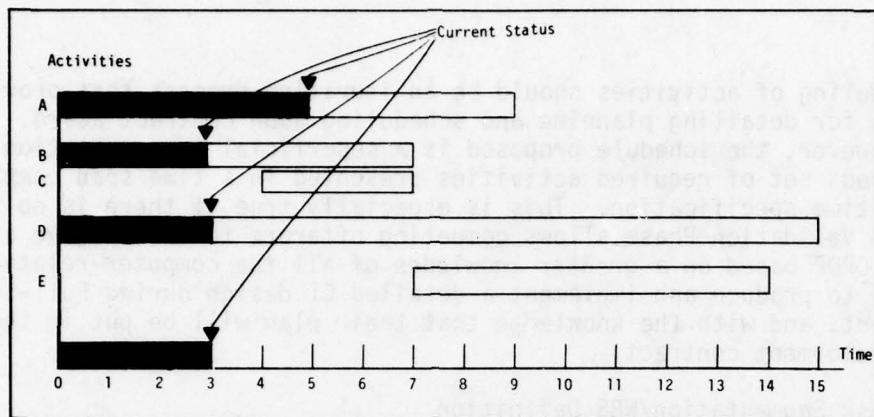


Figure 11. Examples of Gantt, Project Network and Time-Scaled Network Charts. (Adapted from Ref. [21].)

The scheduling of activities should be an iterative process that provides the framework for detailing planning and scheduling upon contract award. All too often, however, the schedule proposed is a superficial representation of the most obvious set of required activities presented in a time span compatible with RFP time specification. This is especially true if there is no Validation Phase. A Validation Phase allows competing offerors to each evolve a detailed SEMP and CPDP based on a greater knowledge of all the computer-related activities necessary to produce and implement a detailed CI design during Full-Scale Development, and with the knowledge that their plan will be put in the Full-Scale Development contract.

3.3.1 Task Segmentation/WBS Definition

During proposal preparation the offeror must correlate his definition of development activities with the SOW which is the source authority for defining activities and against which the Preliminary CWBS corresponds. Generally, one or more WBS elements must be defined for each CI to be used in cost reporting, scheduling, and analysis. Depending upon the size of the system, or the partitioning of the system into numerous CIs, WBS definition should generally be extended below the CI level. (See also Appendix A of Ref. [4].) The extension of WBS elements should be done by the development contractor with PO concurrence.

The offeror's proposed WBS definition of the software should specify the activities involved, and not just define the activity. That is, each WBS element at whatever level should be a stand-alone activity which can be scheduled and costed as part of the total development cost. Further, the WBS definition should represent an orderly flow of activity, providing visibility, control, and cost measurements for each specified activity. For example, in most weapon systems there are several CPCIs identified in the design process (e.g., the Satin IV program defined 68 CPCIs). If the contractor had defined the CPI in the WBS, he might have subdivided activities common to the development of the entire software subsystem and not specific to a single CPI. These activities may include: preparation of the CPDP, design of the CPI interfaces, and development of the program support library. It can be seen that these activities should not be costed under each defined CPI (in Satin IV that would equal 68x3 additional costed activities) because these activities have nothing to do with developing individual CPCIs but rather the entire software system.

The depth of the extended CWBS must be carried to a sufficiently low level to enable the contractor to gain an understanding of the development tasks and establish adequate visibility for both cost estimation and project planning purposes. The level specified should generally be dependent upon the specific proposal, but at the same time, it must directly relate to the activity needed to produce the end item. A helpful consideration in determining the WBS structure is:

Is there an activity in this task which is common to any other task?

When adequate time is spent in the analysis of an activity, there may be a better understanding of the tasks involved. The additional analysis of tasks within an activity provides increased costing accuracy.

The WBS structure as reflected in the CPDP is intended to designate the organization or management structure for the software development effort. During the proposal preparation process it is useful to make an identification of the organization responsible for each task. The effect of compatible WBS activities with the organizational structure is high traceability of resource requirements (who gets what money, why, and when).

3.3.2 Scheduling of WBS Elements

It is important to realize that there is a definite time-phasing, or distribution, of costs for each specific WBS element. The schedule must levy time-phased cost distribution insofar as possible for the specific procurement, since total cost cannot be divided by total time units and since each proposed WBS element has its own resource requirements.

During the course of proposal preparation, the offeror should evolve a schedule for every identified activity (i.e., element) in the WBS structure. Although this is a time consuming and complex set of tasks, it will help in identifying any deficiencies in the WBS structure as well as schedule incompatibilities. Tasks which exist over a long period of time must have resources properly allocated for the entire time (i.e., all identified tasks must be funded for their duration). In addition, the schedule should support preparation of each Contract Data Requirements List (CDRL) item.

SECTION 4 - ROLE OF PARAMETRIC MODELS

4.0 INTRODUCTION

This section briefly discusses the use of parametric models in software cost estimation. In addition, cost variables, used in Cost Estimating Relationships (CERs) are briefly examined. Although the use of parametric modeling for software cost estimation appeals to a large number of personnel both in Government and industry, this method of predicting software development cost has not yet met with a great deal of success. It is the intention of this section to familiarize the reader with the subject of parametric modeling, not to provide an exhaustive and conclusive survey of the technique. A list of currently used parametric models includes:

- RCA's PRICE cost predicting models
- The Putnam model
- The Tecolote model
- The General Research Corporation model
- The Hahn and Stone model
- The GTE-Sylvania method

Three specific examples of this modeling technique are presented in Appendix B.

4.1 PARAMETRIC MODELS

A parametric model for estimating the cost of a proposed software development project is an equation (or group of equations) which expresses a quantifiable relationship of a software project's cost to a number of cost variables. Derivation of the relationship of the cost to cost variables is dependent upon prior collection and analysis of historical cost data and project variables. Using the quantification of the cost/cost variable relationship, new cost estimates may be made by estimating the values of the cost variables representing the new systems and, subsequently, computing an estimated cost by inserting the estimated cost variable rates into the cost estimating equation (i.e., parametric model). Parametric equations allow for relatively fast computation of software cost estimates by varying the estimated cost variable values.

Parametric equations are often computerized, requiring little or no software development experience from their users. However, the equation expressing the relationship of cost variables (i.e., a subset of the project characteristics) to cost depends upon a valid and comprehensive software cost experience basis (i.e., a software cost element data base). Such a data base must contain a large sample of comparable (i.e., consistently defined and collected

as well as unbiased) data. Establishing and maintaining a comprehensive data base of this nature, then, is necessary to develop a parametric model, and requires an in-depth knowledge of the software development process. By use of statistical techniques (e.g., multivariate regression analysis, factor analysis, correlation, and others) an equation is derived which represents the effect of the observed cost variables on the dependent variable. This requires that the analyzers choose the correct independent and dependent variables over a large sample of programming projects. The independent variables selected however, are generally those variables which are easiest to measure or quantify on an absolute scale.

It can be seen, then, that the use of parametric models for estimating software development costs may be performed by a relatively inexperienced software developer. Further, the relationship of project characteristics or cost variables to cost as represented by the parametric model may or may not be valid depending upon the comparability and quality of the data in the data base and upon the statistical analyses on which the model is based. In addition, the independent variables required of the estimator for the model may not be the most important or the only variables impacting development costs. The validity of a model should be demonstrated by its successful use in estimating costs for a number of similar projects. Although a parametric model may be a useful and simple cost estimation technique, it is subject to unreliable estimates and/or biased results. At best the use of a model should be in conjunction with a more comprehensive cost estimation method in which parametric modeling results are combined with detailed analysis of the software's functional performance requirements in a specific work environment.

4.2 MODELS VERSUS METHODS IN COST ESTIMATION

A model purports to describe how and where a software project uses money whereas a method gives directions as to how to produce a cost estimate. One can employ a model to produce the contents of an estimate by attempting to analyze a software project in terms of the model. Or, one can employ a method which gives directions as to the form and use of documents and or supporting material needed for an estimate. Models dictate contents but allow presentation in whatever form is desired; methods dictate form, and force consistency among the several parts of documents, but allow the contents of the cost estimate to vary widely according to the estimator's judgment. Given the current state of the cost estimating art, models tend to oversimplify the problem in order to tackle it, while methods tend to place an overly large burden for tackling the problem onto the cost estimator.

Software cost models are usually of two types, aggregated and disaggregated. In aggregated estimation modeling, the total cost is first derived and then appropriate percentages are allocated to each part of the software development. The allocation may be according to phases, activities, or products. Each part can then be further divided into subparts and costs allocated in

a top-down fashion. In disaggregated estimation modeling, the software project is decomposed top-down as before, but costing is not applied until the total subset of software pieces are of "manageable size". Costing these pieces is presumed to be easier whenever analogous experience with similar completed small pieces can be found. Once the pieces are costed, the total cost is then derived in a bottom-up fashion. Aggregated estimation is by far the easier because it appears to be simpler to allocate costs via percentages than via absolutes. Unfortunately, aggregated estimation does not reveal how the original total cost is to be derived, and this number is generally the most difficult to determine reliably. Disaggregated estimation is more difficult because the process of independently costing each piece resulting from the decomposition seems to require more ability, confidence, and effort in detailing design considerations. However, more confidence can be placed in the results of disaggregated estimation. In practice there is probably a use for both techniques. Aggregated estimation should be used for quick, "seat-of-the-pants" guessing in preliminary planning while disaggregated estimation should be used for formal cost preparation. A cost estimation technique, whether it be a model or a method, must consider constraints and trade-offs inherent in the specific development effort. The major constraints encountered during software development are constraints in hardware (generally from limited memory or limited execution speed), constraints in schedule, and constraints in man-hours. If any of these constraints are stringent, the quality of the software product and/or allocated resources will be impacted. Ideally, a cost estimation technique should be constructed to provide accurate cost estimates of the various trade-offs. Unfortunately, this capability is found in few techniques.

4.3 DEVELOPMENT OF COST ESTIMATING RELATIONSHIPS (CERs)

The cost estimating method presented in Attachment 3 of AFSCM 173-1 outlines the steps for the development of cost estimating relationships. In addition to specifying a methodology, it includes guidance on each of the iterative steps. Because it clarifies the problems inherent in developing CERs, it is summarized below:

- Designate the Dependent Variable. Since cost data for CERs should be collected via the Work Breakdown Structure (WBS), this step includes determining which item at what WBS level should a desired type and quantity of cost data be collected for use in subsequent analysis. Difficulty in deciding the usefulness of a component or functional cost subdivision can be expected.

- Define the Designated Dependent Variable. This task includes precisely defining the designated dependent variable identified in the preceding task. Because of imprecise definitions, differing unit terminology, alternative methods of defining costs, and incompatibility of historical cost data reported by contractors and Government, the definition process is often difficult.
- Select the Parameters to be Tested as Potential Independent Variables in the CER. The selection of technical parameters to be statistically tested should be obtained from expert sources and possess the following qualities:
 - Measurable in quantifiable terms
 - Available for data collection
 - Reflect performance characteristics rather than design characteristics

In addition, when the variables reflect aspects of new technology, extrapolation of costs will be necessary.

- Collect Data on the Independent and Dependent Variables to be Correlated. The data collection task is a lengthy and difficult task which must ensure adequate quantities of comparable, relevant, and high quality data.
- Using Statistical Analysis, Explore Relationships between the Independent and Dependent Variables. This step includes identifying the logical relationship between the dependent variable and one or more independent variables. Various statistical methods of analysis are useful, including scatter diagram association and correlation coefficients, which eventually result in formulating a preliminary functional relationship from one or more candidate relationships.
- Determine the Relationship that Best Describes the Data. This step includes estimating values for parameters in the candidate relationships and choosing the relationship that best describes the data. (Least squares is a useful curve-fitting technique which is readily available in automated form via regression analysis programs.)
- Document the Results. This step includes recording relevant data about the CER in order for that CER to be used reliably by others in costing other systems or resolving problems.

- Validate the Relationship.* This step includes using the model on several projects, collecting data on actual costs and values of independent variables, and then comparing the actual costs with the estimated costs. The model may have to be run with actual values of independent variables if they differ from the values used in the original cost estimation.

*This step is not included in AFSCM 173-1.

SECTION 5 - COST PROPOSAL EVALUATION

5.0 INTRODUCTION

This section provides the PO with evaluation information regarding the software cost proposal. Additional guidance on software contracting can be found in Ref. [5].

Contracting for software acquisition is a complex, interdisciplinary activity, requiring expertise in contracting, finance, law, contract audit, work packaging, engineering, and cost analysis. Statutory requirements concerning the submission of cost data are contained in the Armed Services Procurement Regulation (ASPR), Section III. Regulations governing contracting by DoD are also contained in ASPR.

Generally a contract is awarded to the lowest bidder if his technical proposal meets minimum requirements. Award to the lowest bidder may not be wise or equitable. The low bidder may have grossly underestimated development costs due to inexperience or inaccurate software cost estimation procedures. A low bid may also be a reflection of the offeror's lack of knowledge of the tasks that must be performed. In other cases, a low bid is submitted in anticipation of follow-on contracts. Just because an offeror agrees to develop a product at a given price is no guarantee that the cost incurred during development will equal the price stated. Excessively low bids may force the offeror to make unwise or erroneous management decisions to stay within costs. The procuror should be aware that he is not necessarily protected just because he may have negotiated a Fixed Price contract to develop a product. If the contractor has grossly underestimated the job, he may choose, (or be forced) to default or else to delay delivery, all at no benefit to anyone. The end result for the Government in this case, may be not only the anticipated costs of the contract but also the additional costs incurred as a result of the poor management decisions. (See Ref. [6].) The information in this section is intended to provide a basis for evaluating the offeror's cost proposal in relation to the technical problems inherent in a system acquisition.

5.1 THE MECHANICS OF EVALUATION

An offeror's technical proposal is evaluated by a source selection organization against a set of preestablished technical and management criteria. It is not directly evaluated against other offeror's proposals. Generally, the RFP states the importance of each major criterion against which the offeror's proposal is to be evaluated. Evaluation criteria may include technical, management, and cost factors. The technical proposal evaluation process culminates in a numerical rating of each proposal, with at least some delineation of the significant advantages, disadvantages, and risks of each proposal.

The cost proposal is generally not included in this rating and is usually evaluated separately by a team of cost evaluators, often outside the PO. The purpose of separating the cost proposal from the technical proposal is to avoid inept evaluation of the dollar relationship to the effort.

In tracking the software cost estimate derived, first, by Air Force personnel and, second, by offerors responding to an RFP, many complexities and problems are identified. The numerous program-dependent variables in all development projects require the examination of many factors to realistically estimate costs. Air Force evaluators must therefore be able to evaluate the labor and materiel components of the offeror's cost proposal and delivery schedule within PO-established constraints. The information needed to make such evaluations should be accumulated and refined by PO Program Control prior to receipt of the offeror's cost proposal. This information forms the basis of the Government's pricing objective and it should include cost and price analyses, historical cost/price data, independent Government cost estimates, and economic analyses. Its comprehensiveness can determine how thorough an analysis of the offeror's cost proposal can be made, and thus can impact the eventual success of the contract award.

5.2 COST ANALYSIS OF TECHNICAL ACTIVITIES

Cost analysis is the review and evaluation of the offeror's cost data with respect to the technical proposal to determine its reasonableness and accuracy. This may be ascertained by examining the tasks proposed to develop each product identified as a deliverable. The cost analyst must then coordinate his examination of the offeror's cost and schedule data with the analysis of the offeror's technical proposal in order to determine the accuracy of the cost proposal. The schedule proposed by the offeror must reflect the allocation of time to product development. Using the commonly accepted 40-20-40 rule (40 percent analysis/design, 20 percent code, 40 percent testing), or other rules of thumb, the cost analyst should examine the allocation of time to each task to determine if the schedule is reasonable. Another factor to consider in appraising the work packaging vs resource and time allocation is the comparative difficulty of the different work elements. Some CPCIs, such as a control or operating system, may be more complex and require more manpower for development. When the offeror has separated and costed each task and subtask, the cost analyst is better able to determine the reasonableness of resource and time allocations, especially for complex applications.

There are a number of support tasks required by every project to produce deliverable product(s). These support tasks may include:

- Design reviews
- Computer program support library development
- Test planning for development, qualification, and integration testing.

- Simulation model development to validate design concepts and approaches.
- Development of a test bed for simulation and testing

As he did for each deliverable product, the cost analyst should determine if the specification of these tasks is complete and if the cost and schedule data presented in the proposal reasonably reflect their performance.

Cost analysis also includes verification of cost data, evaluation of specific elements of cost, and projection of costs to determine their overall effect on the program. The latter includes examination of the total cost proposed by the offeror in performance of all tasks specified in the SOW, including:

- Direct costs, including labor costs (cost of performing all tasks), travel and relocation costs, computer time, CDRL pricing.
- Indirect costs.
- Contingency costs.
- General and Administrative (G&A) costs.
- Fee.

The basic direct labor costs can be verified by examination of the offeror's tasking of explicit, derived, and implied customer requirements. These tasks include all those necessary for generation and delivery of all CPCIs, and required support software, related documentation, and all other deliverable data specified in the CDRL. The total set of tasks presented by the offeror indicates his understanding of the work to be performed and the labor required. The number and diversity of tasks for each software development will vary, depending both on the requirements and the extent of the contractor's responsibility. For example, a prime contractor for a system/system segment will be responsible for the complete set of development tasks, while a subcontractor for CIs/CPCIs of a system segment will be responsible for a delineated set of tasks. In addition, each software development effort has unique requirements which complicate the evaluation process. However, the time spent in examining the offeror's specification of work packages can lead to the rejection of the offeror who does not understand the technical details of the software system.

Generally, the cost proposal does not elaborate on the cost estimation method used by the offeror, although sometimes the total number of instructions (in either source or object code) is given. Some, or all, of the factors discussed in Section 3 should have been taken into account by the offeror in the derivation of the total number of man-months needed to perform the tasks.

Generally, little data is provided in the cost proposal to justify the estimate of total man-months necessary for task performance. Thus, examination of the detailing of tasks and subtasks in the schedule must be the principal method for determining the rationale of the proposed costs. If the offeror has included estimates of the size of each CPCI (possibly to the CPC level if a Validation Phase has been conducted and this source selection will determine the Full-Scale Development Phase contractor), they should be compared to the Government's cost estimate. Depending on the level of confidence in the in-house estimate, the comparison of estimated sizes might provide a good indication of the offeror's rationale for proposed labor costs. However, because software size estimates are notoriously inaccurate, comparisons of size estimates cannot in themselves validate the offeror's understanding of the problem. Further, the number of instructions is only one project-dependent variable that impacts cost.

Once the offeror has established the man-month requirements, the designation of the type of man-month necessary for each labor-related task must be verified. Insufficient project planning may be recognizable if the offeror has not allocated specific skill levels for specific tasks. For example, analysis and design activities require software engineering support (or the equivalent) whereas module programming and testing require a less experienced, and therefore, less expensive allocation of labor.

Labor costs associated with document production are more accountable than labor costs associated with task performance because documentation tasks include all activities associated with producing contractually-specified documentation. DoD requirements for documentation are specified in the Contract Data Requirements List (CDRL) and are generally also specified in the corresponding SOW. It includes all documents specified in the SOW task description, and may also include other documents not mentioned in the SOW. The costing of CDRL items is the dollar value required for each CDRL item, and generally includes the combined costs of all labor and document reproduction associated with the task. Examination of the data presented in the cost proposal for consistency with the SOW/RFP is a necessity during proposal evaluation.

Other direct costs include travel allocations, generally for temporary duty trips and relocations. Such costs must necessarily include information relating the trip to a specific task, the number and duration of trips, and per diem rates.

Computer utilization is another direct cost which may significantly impact total estimated project cost. It is important to ascertain the reasonableness of estimated computer utilization scheduling. Computer costs are obviously time-phased and should correspond to the tasking of code and test activities of support or application programs unless special tasks require computer resources. An example of special tasking is the use of simulation programs intended to verify design alternatives or timing/sizing studies. The estimated size of the total development software package should be considered in relation to the total amount of computer resources costed.

Because computer utilization requirements vary, as do rules for costing them, it is not always feasible to determine the offeror's interpretation of the computer resources needed to perform the development job. Computer utilization resources may include customer-furnished equipment, purchased or leased equipment, customer-furnished computer time, and purchased computer time. The cost of the other sources of computer time is generally made on the basis of other costs, as follows:

- Customer-Furnished Equipment. Cost of computer operations, including computer operators, maintenance, facilities, and air conditioning.
- Purchased or Leased Equipment. Cost of purchase or lease and cost of computer operations.
- Customer-Furnished Computer Time. No cost.

Another direct cost in computer utilization estimates is derived from the necessity to remotely access the computer or to use a timesharing processing mode instead of batch. Both these costs appear to increase the total computer costs, although indirect benefits are derived from their usage as demonstrated by an increase in programmer productivity.

Because of the variability associated with computer utilization (i.e., computer execution times, application differences, operating system differences), no standard algorithms for justifying computer utilization can yet be provided to aid in the analysis of estimated costs.

Cost proposals are often assembled late in the proposal production process by the bidders and contain discrepancies when compared to the technical and management proposals. The cost analyst should especially look for:

- All products and tasks, especially CPDP tasks reflected in the WBS, schedule, and proposed costs.
- The correspondence between manpower loading and the labor rates and levels bid.
- Requirements for Government-furnished property, computer time, or labor that are beyond the scope of the RFP or that may provide unfair competitive advantage.

Justification of indirect costs is also a task in the analysis of the cost proposal. Indirect costs generally include management and overhead costs and the basis for all rates must be examined. The procedures and regulations for analysis and evaluation of this type of cost are presented in ASPR.

5.3 COST ANALYSIS OF TECHNICAL/FINANCIAL MONITORING ACTIVITIES

A contractor is required, in accordance with DoD Instructions 7000.2 and 7000.10, to submit summary-level cost, schedule, and performance data to provide the PO with a means for evaluating deviations from planned costs and schedules. Therefore, an offeror must provide in his proposal sufficient data about his management, performance, cost, and schedule reporting system to assure that it is compatible with the information requirements of the Government's Cost/Schedule Control Systems Criteria (C/SCSC). These criteria are presented in AFSCP/AFLCP 173-5, and provide guidance for the uniform planning and implementation of cost/schedule reporting and surveillance of contractor compliance. The Cost Performance Report (CPR) is the reporting vehicle used, and its effectivity depends upon the WBS definition. WBS identifiers provide the bases for tracking actual technical progress within costs and schedules against the performance estimates for each WBS element.

The WBS forms the framework by which the contract work statement tasks, contract line items, Configuration Items (CIs), contract specification tree, and the offeror's response to the RFP will be correlated. After contract negotiations, which may result in adjustment to the WBS definition, the WBS becomes the basis for further extensions during Full-Scale Development/Production. Although the offeror has complete flexibility in extending the WBS to reflect his work packages, the summary elements' nomenclature and definitions may not be changed after contract award.

The C/SCSC requires that the contractor report cost and schedule data on a monthly basis for each identified WBS element. In addition, he must report schedule and cost variances when cost/schedule thresholds are exceeded and explain the corrective action being taken to resolve the variance. Using the WBS element for cost and schedule reporting allows the PO to sum all costs associated with each Level 1, 2, or 3 element to ascertain if total costs exceed thresholds for that element. Small variances reported by the contractor may have a cumulative effect that could result in significant budget overruns for the WBS element.

The WBS allows contractor flexibility in work packaging, cost reporting, and performance monitoring, but because of its flexibility, cost data received by the PO is often aggregated. For example, the WBS for COMBAT GRANDE provided for the submittal of cost information at the combined CPCI level, which did not include a breakdown of system engineering and management activities and associated costs. The consequence of this aggregated cost reporting process is the loss of valuable historical information that can never again be obtained because of its proprietary nature. Unless the WBS definition is correctly extended and becomes the contractual vehicle for cost reporting, adequate cost data for future reference will not be provided to the PO. To collect costs at a more meaningful level, the PO must require that the WBS definition be extended to the CPCI-level at a minimum. Another recommendation

for more thorough data collection and subsequent program control includes the submission of manpower and dollar costs broken out by phases of the development process (i.e., design, code, module test, CPC test, CPCI test, and qualification testing) for each CPCI (see Appendix A of Ref. [4]).

Monitoring technical progress may be accomplished more easily if valid technical milestones are required for each CPCI. Sometimes, the technical status of software development is derived by examination of the percent of resources expended. However, resource expenditures and work progress do not generally follow parallel curves to project end. When a variance exists between budgeted cost for work scheduled, budgeted cost for work performed, and actual costs, the contractor must account for the variance. When actual performance deviates from planned performance, the contractor's management control system should be required to trace the variance to its source.

To obtain valid and meaningful data on technical progress, while ensuring that the data obtained is meaningful for further acquisitions, the WBS definitions must be extended to more adequately identify and collect software development costs while providing for technical milestones within each CPCI.*

The Program Breakdown Structure (PBS) supplements the use of the WBS for programs managed by AFSC when prescribed by PMD or AFSC Form 56. The Program Breakdown Codes (PBCs) provide an effective identification mechanism to be used in support of uniform cost accounting for comparisons across systems. AFSCM 173-4 provides policies and procedures for assigning PBCs to WBS elements for AFSC-managed programs. With regard to software cost reporting, the PBS is intended to provide an automatic reporting system for the collection of cost data and a mechanism for reflecting all costs in a program. Further discussion of the PBS and PBCs is presented in Appendix A of Ref. [4]. In addition, the initial expansion of PBCs presented in the appendix are further expanded and modified to fit into the cost reporting system proposed by General Research Corporation in Volume 1 of Ref. [22].

The following guidelines have been recommended (see Ref. [1]) to ensure that the software development costs for an embedded computer system are properly reflected in the WBS:

- A single element in the WBS for software development seldom accounts for the total software development cost. Usually the single element accounts only for coding and checkout costs, which is generally 20 percent of the total software development effort.

*The MITRE Corporation is developing and is scheduled to deliver to ESD by mid-1978, a new Data Item Description to be used to implement a standard cost reporting system for software acquisition.

- Analysis, design, testing, and integration should be reflected in the WBS. Since the WBS for most systems with embedded computers is oriented around prime mission equipment elements, the portions of each prime mission equipment element targeted for software implementation must have separate software elements for analysis, design, testing, and integration.
- Management and support costs for software development should be adequately reflected in the WBS. This may be accomplished by including software elements in the system engineering project management portion of the WBS. In addition, these elements should be partitioned by the software life cycle phases.
- Separating or factoring hardware elements from software elements in a satisfactory manner in the WBS may be difficult for many developments. Engineers, especially in avionics applications, may be qualified in both hardware and software, and partitioning their time accurately among the various WBS elements may be difficult, especially in the testing and integration phase. Problems encountered then cannot be attributed to hardware or software until the cause has been found and the problem is resolved. The only solution for accountability of hardware/software costs may be constant watchdogging to ensure that labor gets partitioned accurately among hardware and software elements in the WBS.*

*It is important that the WBS reflects the way work is performed. For example, system-level tasks such as system integration and problem isolation should be classified and reported at the system level and should not be allocated to software until software work is obviously performed. Artificial breakdowns will be recognized as such and will be arbitrarily reported, providing useless information.

APPENDIX A - REFERENCES

1. "Software Cost Estimation Study, Volume II: Guidelines for Improved Software Cost Estimation"; RADC TR 77-220; Doty Associates, Inc.; Rockville, MD., February 1977.
2. AFSCP 800-3; "A Guide for Program Management"; April 1976.
3. MIL-STD-881A; "Work Breakdown Structures for Defense Materiel Items", April 1975.
4. "Software Acquisition Management Guidebook: Statement of Work Preparation;" Glore, J. B., and Bjerstadt, W. R., MITRE; ESD-TR-77-16; January 1977.
5. "An Air Force Guide to Contracting for Software Acquisition"; Bolen, N.E., MITRE Corp., ESD-TR-75-365; January 1976.
6. "An Exploratory Study of Software Cost Estimating at the ESD;" Devenny, T.J.; GSM/SM/76S-4; Thesis, AF Institute of Technology, Air University; July 1976.
7. "The Management of Computer Programming Projects;" Lecht, Charles; American Management Associates, Inc., 1976.
8. "The Cost of Developing Large-Scale Software"; Wolverton, R. W.; IEEE Transactions on Computers; Volume C-23, No. 6, pp 615-636; June 1974.
9. "Estimating Resources for Large Programming Systems"; Aron, J.D., IBM Federal Systems Center, 1969.
10. "Estimating the Costs of Programming Systems"; Meyers, G.J.; IBM Technical Report, TR 00.2316; May 1972.
11. "Research into the Management of Computer Programming: A Transitional Analysis of Cost Reporting Techniques". Weinwurm, G.E., Zagorski, H., ESD-TR-65-575, November 1965.
12. "Factors that Affect the Cost of Computer Programming;" Farr, L., Nanus, B., SDC., June 1964.
13. "Research into the Management of Computer Programming: A Quantitative Analysis"; Farr, L., Zagorski, H., SDC, January 1965.
14. A Summary of an Analysis of Computer Programming Cost Factors", Farr, L., Zagorski, H., SDC, January 1965.
15. "An Air Force ADP Experience Handbook (Pilot Version)", Gradwohl, A.J., et al, ESD-TR-66-673; December 1966.

16. "An Air Force Guide for Monitoring and Reporting Software Development Status"; ESD-TR-75-85, MITRE; September 1975.
17. "The Mythical Man-Month", Brooks, Frederick, P., Jr., Addison-Wesley, 1975.
18. "An Air Force Guide to Computer Program Configuration Management"; Searle, L.V., SDC, ESD-TR-77-254, August 1977.
19. "An Air Force Guide to Software Documentation Requirements"; Schoeffel, W.L., MITRE, ESD-TR-76-159, June 1976.
20. "A General Solution to the Software Sizing and Estimating Problem"; Putnam, Col. L. H., U.S. Army Computer Systems Command.
21. "Project Management with CPM and PERT"; Moder, Joseph, and Phillips, Cecil R., Van Nostrand, 1970.
22. "Cost Reporting Elements and Activity Cost Tradeoffs for Defense System Software, Volume I: Study Results"; Graver, C. A., et al.,; CR-1-721; General Research Corp; March 1977.
23. "An Exploratory Investigation of Programmer Performance Under On-Line and Off-Line Conditions"; Grant, E.E., and Sackman, H., IEEE Transactions on Human Factors in Electronics, March 1967.
24. "Software Transfer Cost Estimation Technique"; Hahn, W. and Stone, J. Jr., M70-43; MITRE, July 1970.
25. "When Should You Emulate"? Lichtenstein, H.A., Datamation, November 1969.
26. "Program Conversion"; Kahn, P.G., Fuller, M.E., Data Processing Magazine, November 1969.
27. "A General Solution to the Software Sizing and Estimating Problem"; Putnam, Lawrence H., presented at Life Cycle Management Conference; AIIE; February 1977.
28. "A Macro-Estimating Methodology for Software Development"; Putnam, Lawrence H., Digest of Papers, Fall COMPCOM '76; pp 138-143; September 1976.
29. "The Influence of the Time-Difficulty Factor In Large Scale Software Development"; Putnam, Lawrence H., Digest of Papers, Fall COMPCON '77; September 1977.
30. "Useful Tools for Project Management"; Norden, Peter V., Management of Production, Penguin Books, 1970.
31. "A Provisional Model for Estimating Computer Program Development Costs"; Frederic, Brad C., Tecolote Research Inc., December 1974.

32. "Estimation of Resources for Computer Programming Projects;" Morin, L., M-5222 Masters Thesis; University of North Carolina at Chapel Hill; 1973.
33. "Summary Notes of a Government/Industry Software Sizing and Costing Workshop;" Geran, D.B.; USAF(ESD); Bedford, Mass; October 1974.
34. "Management Handbook for the Estimation of Computer Programming Costs;" TM-3225/000/01; System Development Corporation; Santa Monica, Calif.; March 1967.
35. "Computer Resource Requirements for Programming Development;" Aron, J.D, Arthur, R.W.; IBM; Gaithersburg, Maryland; 1975.
36. "Factors in Software Quality;" McCall, J.A., Richards, P.K., Walters, G.F.; Information Systems Programs, General Electric Company; Sunnyvale, Calif; October 1976.
37. "Software Acquisition Management Guidebook: Software Quality Assurance;" Neil, George; SDC; ESD-TR-77-255; August 1977.

APPENDIX B - SOFTWARE COST ESTIMATION MODELS

1.0 INTRODUCTION

This appendix presents three parametric models as examples of the types currently being used in some specific software cost estimation problems*. Although each model presents estimating relationships according to the type of software application problem being addressed, the three models are quite different in their approach. The models presented include:

- A method for estimating costs involved in transferring software from a source computer to a target computer.
- A method for estimating cost among divisions of time during which software development is proceeding.
- A method for estimating cost relationships for a tactical fire control development effort.

1.1 HAHN & STONE SOFTWARE TRANSFER COST ESTIMATION TECHNIQUE

The Hahn and Stone technique uses a model designed to assist in the estimation of the cost of transferring software from one computer to another. This technique was developed by the MITRE Corporation in response to a request by the Defense Communications Agency to undertake a study of the costs involved in transferring a set of applications software from one computer system to another (see Ref. [24]).

1.1.1 Factors Considered in the Hahn & Stone Model

Much of the paper presenting the Hahn and Stone technique is a discussion of the four major factors determining how the software can be transferred, and consequently, the costs necessary for the transfer.

Unlike most software cost estimation problems, the costs of transferring a software system to a computer different from the one on which the software was developed generally involves estimating costs for many fewer tasks than were necessary for the original software development. The entire requirements analysis and, usually, the design phases of the software life cycle are unnecessary. Unless the software conversion requires redesign, the software cost estimation problem is greatly reduced because many of the factors involved in the task of estimating, such as those discussed in Section 3 of this guidebook, have been removed or are more quantifiable. For example, both the software size and the complexity of the application are directly observable via the existing software documentation. (This does not mean to imply, however, that total conversion costs will be significantly less than the total development costs. It does imply, however, that much of the uncertainty in the cost estimate is removed.) By omitting requirements analysis and design tasks, the costs to be estimated depend upon the nature of the programming and testing activities. In order to evaluate the manpower required for these tasks, Hahn and Stone consider the impact of hardware characteristics (both source and target), the nature of the software to be transferred, the transfer techniques available, and the transfer phasing plan strategy.

*The models are presented for illustration only. This does not constitute an Air Force approval of these models.

In general the greater the number of hardware differences between the target and source computers, the greater the amount of manpower required to resolve the differences in the transfer of software. Hardware characteristics which should be expected to impact transferability of software include differences in transfer rates of instructions, changes in word size, loss of bit significance, changes in system-dependent parameters that control literal items, machine configurations of a floating point number, and machine-dependent constants. The software characteristics which effect conversion costs include the size of the software, programming languages, and machine-dependent characteristics of the software.

The H&S Model is dependent on software transfer techniques which are classified into three types, (1) direct use, (2) machine-assisted transfer, and (3) manual program transfer. These types of transfer techniques directly impact the amount of manpower resources required to transfer the software to the target machine. These considerations are summarized in Figure 12.

Direct use implies that the object programs are directly transferable because the source and target computer are highly compatible, or the target computer can be made to emulate the source computer's instruction set by hardware or simulate it by software. The cost of emulation hardware is estimated to range between \$1,200 and \$9,000 per year (see Ref. [25]). The cost of development for a simulator is estimated to range from \$40,000 to \$70,000, and require 4 to 9 months to develop (see Ref. [26]). Since these costs are not related to any system size by Hahn and Stone, and do not include costs of the inefficient use of the computer for both emulation and simulation, their validity is suspect.

Machine-assisted transfer techniques employ a language processor to aid in the conversion, such as a decompiler, translator, and compiler. Machine-assisted transfer techniques almost always assume that a one-for-one relationship can be established between the source and target languages. They require that the developer thoroughly understand the source and target languages and that the current program source deck be available and essentially free of machine-dependent functions or data descriptions. Machine-assisted techniques will not improve the efficiency of the source program. The only improvements to be expected are those attributable to the difference in the hardware and the operating system.

Manual program transfer techniques are totally dependent on human effort and may consist of three types, redesign, reprogramming, and recoding. They require the largest amount of resources. Most software conversion projects will require at least some manual conversion tasks. The amount is determined by the following factors:

- The degree to which machine-assisted translation techniques are not available, or the amount of instructions remaining after the use of a machine-assisted transfer technique.

TYPE	CHARACTERISTIC	IMPACT
Hardware Characteristics	Change in word size	May require table repacking if target computer has larger (and storage efficiency is required) or smaller word size.
	Loss of Significance	A reduction in the number of bits contained in an integer may cause loss of significance in the value or less precise arithmetic operations in the target computer may cause reprogramming.
	Literal Items	A change in system parameters controlling compiler-generated literals (especially bytes/word, bits/byte) will increase size and complexity of transfer.
	Floating Point	Target computer's configuration of a floating point item may have impact on transfer.
	Machine-Dependent Constants	Machine-dependent constants (i.e., hexadecimal, octal) may need reformatting or integer constant word size may need changing.
Software Characteristics	Size	Larger programs appear to require a disproportionately large amount of resources. In addition, data base size may impact timing requirements, which may become a cost factor.
	Program Languages	Transfer of a Machine-Oriented Language (MOL) system is dependent on differences between source and target computers. Also, computer-aided MOL transfer techniques are not readily available. Transfer of an HOL system is generally less expensive, assuming that the HOL compiler is available on the target computer.

Figure 12. Software Transfer Considerations (Adapted from Ref. [24]).

TYPE	CHARACTERISTIC	IMPACT
Software Characteristics (cont'd)	Other Software Characteristics	Status and quality of documentation. Number of subprograms. Number and complexity of linkages between OS/Programs. Interfaces between programs and data base. Number of modifications to be made during transfer. Growth rate of programs and files. Data storage media.
Program Transfer Techniques	Direct Use - Modification of the target computer (i.e., emulation, simulation)	Most useful for programs with short life span or infrequent use. Full capabilities of target computer may not be used.
	Machine-Assisted Transfer	<u>Decompilation</u> - The process of translating machine language to a compiler language. Effective decompilers are feasible but rare; idiomatic machine-language expressions and code that modifies itself complicate the decompilation process. <u>Translation</u> - The process of converting a program in one language to a program in a different language. Requires correct usage of MOL/HOL constructs. <u>Recompilation</u> - The process of compiling a POL source program on the target computer with the same POL compiler.
	Manual Transfer	<u>Redesign</u> - The process of producing a different problem solution and processing logic than original source. Makes maximum use of target computer and support software. Most costly.

Figure 12. Software Transfer Considerations (cont'd)

TYPE	CHARACTERISTIC	IMPACT
Program Transfer Techniques (cont'd)	Manual Transfer (cont'd)	<p><u>Reprogramming</u> - The process of producing a target program that performs the same function as original source, but may use different logic. Used when existing program has been extensively retrofitted, efficiency needs improvement, or interface with equipment has changed.</p> <p><u>Recode</u> - The process of manually translating source language program to target language program. Least costly.</p>

Figure 12. Software Transfer Considerations (cont'd)

- The amount the programs have undergone extensive or piecemeal modifications since initial development and require redesign for more efficient operation.
- The use of idiomatic expressions* and program self-modification make machine-assisted translation ineffective.
- A decision is made to convert MOL programs to POL programs in order to achieve more inter-computer compatibility.
- The degree to which existing programs are combined to achieve more efficient operations.

Redesign is used to produce a target program which has a different problem solution and different logic than the source program. It is usually the most expensive of the manual techniques since it normally includes all tasks associated with the original development of a program. (See Figure 13.) However, it may be advantageous if two or more programs are combined or extensive modifications have changed the original function of the program. Program redesign is also necessary if the mode of operation is to change; i.e., batch mode to on-line mode.

Reprogramming is used to produce a target program which performs the same function as the source program but which is not constrained by the use of all of the original logic. This method would most likely be applied when the existing program has been extensively "patched", a substantial increase in efficiency is desired; or major changes in the use of peripheral equipment are made (change from tape storage to disc storage). The tasks involved in reprogramming are shown in Figure 13.

1.1.2 Hahn and Stone Model**

The Hahn and Stone cost estimation model considers the cost of transferring programs (C_p), the cost of transferring data (C_D), and other costs (C_0), such as documenting programs and keypunching. These costs are all expressed in man-years***. Computer costs are expressed in terms of hours of computer time required for software transfer. The model for estimating these costs considers the use of various combinations of transfer techniques, and program characteristics. The cost estimation model is represented by the following equation:

*An idiomatic expression is a program segment of one or more instructions whose meaning is not a direct result of the meaning of the individual instructions. Idiomatic expressions occur often in machine-oriented languages.

**Based on actual development and modification costs of programs, with further refinement by DoD, industry, and MITRE personnel.

***The conventional values used for man-time conversions are: 1 man-year = 12 man-months = 260 man-days = 2080 man-hours and 1 man-month = 173.3 man-hours.

TASK	REDESIGN	REPROGRAM	RECODE
1. <u>Analyze System Requirements</u> - Determine the operational requirements of the system; evaluate their completeness, feasibility, and compatibility with other systems. Analyze the operational and user needs for output and sources for inputs.	X		
2. <u>Analyze Program Requirements</u> - Determine the requirements for program production and test, program language to be used, operating system and other programming support required.	X	X	
3. <u>Analyze Similar and Interfacing Systems</u> - Determine the systems procedures and techniques already in production, operation, or planned which may influence the redesign of the system.	X		
4. <u>Prepare Design Requirements and Specifications</u> - Develop the specifications for the total data processing system including the hardware configuration required. Develop the design requirements for the software system including programs, data structure and interfaces required to satisfy the operational requirements.	X		
5. <u>Design Program</u> - Using the design requirements or existing program documentation, design the entire computer program system and/or individual programs and routines that have been identified. Determine and design data input and output formats.	X	X	
6. (a) <u>Code the Program</u> - Translate flow diagrams and other statements of program design into coded instructions.	X	X	
(b) <u>Recode the Program</u> - Translate the code, using program listings and other program documentation from the source to the target language.			X
7. <u>Desk Check the Program</u> - Desk check the new code by looking for illegal expressions (which may occur when idiomatic expressions are used in the old program), erroneous data references (which may occur when source program is self-modifying), program inefficiencies, and deviations from program specifications.	X	X	X

Figure 13. Manual Transfer Tasks

TASK	REDESIGN	REPROGRAM	RECODE
8. <u>Compile and Check Program Code</u> - Assemble or compile each program into machine-readable form, check listings for errors, correct code, and reassemble or recompile. Continue the process until a satisfactory program is obtained.	X	X	X
9. <u>Test Individual Programs</u> - Within requirements of the program test plan, run performance tests of individual programs to isolate and correct errors. Rerun until the requirements for problem solution are satisfied.	X	X	X
10. <u>System Integration and Test</u> - Run the program system test for physical integration of functionally independent programs to isolate and correct failures to meet the program system requirements (Program system consisting of only one individual program will not require this task).	X	X	X

Figure 13. Manual Transfer Tasks (cont'd)

$$(1) \quad C_T = C_P + C_D + C_O$$

where: C_T = Total Cost

Each term of the equation and its derivation is discussed below.

The cost of program transfer (C_P) is made up of two major components: the cost of the machine-assisted transfer techniques and the cost of manually transferring all or part of a program. Expressed as an equation:

$$(2) \quad C_P = C_A + C_M$$

where: C_P = Cost of individual program transfer

C_A = Cost of the machine-assisted transfer technique. (No attempt is made to further quantify or break down this cost element. However, machine-assisted transfer is discussed in general from a technical and cost point of view.)

C_M = Cost of manual transfer. [The cost of manual transfer of the programs is the largest single cost of the transfer process and consists of: (1) the number of instructions which must be manually transferred; (2) the average rate at which these instructions can be transferred, expressed in instructions/man-day and; (3) the cost of manpower per man-day.]

Expressed as an equation:

$$(3) \quad C_M = \frac{I}{R_T} (C_{MD})$$

where: I = Number of instructions to be manually transferred (dependent on size of program, the transfer technique used, and the effectiveness of the machine-assisted transfer.)

R_T = Rate of transfer. (The average number of instructions or statements which can be manually transferred from one computer to another in one man-day.)

$$(4) \quad R_T = \frac{I}{MD_T}$$

where: MD_T = Total number of man-days required

Further, MD_T can be defined as follows:

$$(5) \quad MD_T = \frac{I}{R_{BC}} + \left[(D_{F1}) \left(\frac{I}{R_{BC}} \right) \right] + \left[(D_{F2}) \left(\frac{I}{R_{BC}} \right) \right] + \frac{I}{R_{BT}} + \left[(D_{F3}) \left(\frac{I}{R_{BT}} \right) \right]$$

where: R_{BC} = baseline conversion rate

R_{BT} = baseline test production rate

D_{F1} = documentation degradation factor

D_{F2} = program instability degradation factor

D_{F3} = system integration degradation factor and further:

$\frac{I}{R_{BC}}$ = number of man-days required for baseline conversion

$(D_{F1}) \left(\frac{I}{R_{BC}} \right)$ = number of man-days required to make up for document degradation factor.

$(D_{F2}) \left(\frac{I}{R_{BC}} \right)$ = number of man-days required to make up for program instability degradation factor.

$\frac{I}{R_{BT}}$ = number of man-days required for baseline test

$(D_{F3}) \left(\frac{I}{R_{BT}} \right)$ = number of man-days required to make up for system integration degradation factor.

Recoding is a one-for-one manual translation of instructions from a source language to a target language. It produces an operational program on the target computer using the same logic and same problem solution as was used in the source computer program. Although Hahn and Stone claim that recoding represents the least sophisticated of the manual conversion techniques because the programmer is not required to have an in-depth understanding of the program logic or solution, the author is doubtful that such a program can ever be thoroughly tested and debugged unless such an understanding exists. The tasks involved in recoding are shown in Figure 13.

Once the size of the program has been determined, the percentage of instructions/statements to be manually transferred can be identified. Because this technique requires the most manpower, it is the principal cost factor and is in direct proportion to the type and effectiveness of the transfer technique used. Direct use transfer techniques (emulation and simulation) are principally

techniques for time phasing the transfer and thus are not considered by Hahn and Stone as part of the direct costs of transferring programs. Machine-assisted transfer techniques are also considered in estimating the transfer cost since the number of residual instructions which remain to be manually transferred is in proportion to the effectiveness of these techniques. The majority of the information presented by Hahn and Stone expressed the cost of transfer, using the machine-assisted technique, in terms of a range of percentages of original program development costs. This method of calculating the cost of using a particular machine-assisted transfer technique may not be feasible or may be subject to error because the original program development cost is often unknown or unreliable.

Development production rates for a total program were then calculated from a sample of actual production rates for each language. These were further refined (see paragraph 1.1.3) to determine baseline conversion and test production rates (R_{BC} and R_{BT}) which are shown in Figure 14. The high and low rates were calculated in that the probability of an actual production rate falling outside these limits was 2.5 percent.

In addition to production rates, degradation factors were introduced to account for the quality of documentation (D_{F1}), the number and magnitude of program modifications required during program transfer (D_{F2}), and program complexity (D_{F3}).

By incorporating equation (5) into equation (4), the expression for R_T becomes:

$$(6) \quad R_T = \frac{(R_{BC}) (R_{BT})}{(1+D_{F3})R_{BC} + (1+D_{F1} + D_{F2}) R_{BT}}$$

The cost of transferring data (C_D) is small when compared to the cost of transferring programs. Data transfer costs include computer time and, if not supplied by the equipment vendor, the cost of a function-specific program to transfer the data. However, if purging of the data in the files or the establishment of new files during the conversion are required to improve operating efficiency and effectiveness, considerable manpower may be required for the development and definition of the data elements, layout of storage allocations, and the data structure.

Other costs (C_0) can be expected during software transfer. These include key-punching and verifying, training, facilities, planning, and management.

The Hahn and Stone cost model can thus be summarized as follows:

$$C_T = \sum I \left[\frac{(1+D_{F3})R_{BC} + (1+D_{F1} + D_{F2}) R_{BT}}{R_{BC} R_{BT}} \right] C_{MD} + \sum C_A + C_D + \sum C_0$$

Where the first and second summation symbols represents sum over all the programs in the inventory and the third summation symbol represents sum over all the other costs.

LANGUAGE		REDESIGN (R_{BC})	REPROGRAM (R_{BC})	RECODE (R_{BC})	PROGRAM TEST (R_{BT})
FORTRAN	Mean	8.2	11.3	22.5	14.1
	Low	6.0	8.3	16.5	10.4
	High	10.4	14.3	28.5	17.9
COBOL	Mean	10.5	14.5	29.0	18.3
	Low	8.2	11.3	22.5	14.1
	High	13.1	18.0	36.0	22.6
JOVIAL	Mean	12.5	17.3	34.5	21.7
	Low	7.3	10.0	20.0	12.6
	High	17.8	24.5	49.0	30.8
MOL	Mean	14.7	20.3	40.5	25.4
	Low	12.0	16.5	33.0	20.7
	High	17.6	24.3	48.5	30.4

Figure 14. Conversion Production Rates (Instructions or Statements/Man-Day)

1.1.3 Summary Evaluation of Hahn & Stone Software Transfer Cost Estimation Technique

The Hahn and Stone technique is designed to assist in estimating the costs of transferring software from one computer system to another. After identifying a number of factors, it provides a rationale for how these factors impact cost. It then presents an aggregated cost model reflecting these factors as well as tabulated production rates for instruction transfer and degradation factors for documentation quality, program stability, and system integration.

The main accomplishment of the Hahn and Stone technique is not the model but rather the tabulation of production rates and the development of the degradation factors used in working the model. The data used to compute these figures was gathered from some 74 computer programs in four languages, including 29 FORTRAN, 8 COBOL, 18 JOVIAL, and 19 MOL programs.

Assuming that the development production rate for computer programs follows a normal distribution and that the program sample chosen was representative, the mean and standard deviation for each language was calculated. Using the Student -t distribution, the higher and lower production rates were calculated so that the probability of an actual average production rate falling outside these upper and lower limits was 2.5 percent. The resulting calculations are shown in Figure 15.

LANGUAGE	LOW	MEAN	HIGH
FORTRAN (N=29)	3.3	4.5	5.7
COBOL (N=8)	4.5	5.8	7.2
JOVIAL (N=18)	4.0	5.7	7.4
MOL (N=19)	6.6	8.1	9.7

*where N = sample size

Figure 15. Development Production Rates (Instructions/Man-Day)

The rates in Figure 15 represent production rates for the total development of a program. However, since conversion of an existing program does not include all the functions normally associated with a development project, the development rates were modified to include only those functions associated with recoding, reprogramming, or redesign. This, then, established a baseline conversion production rate (R_{BC} in the model). In addition, they developed a program testing production rate (R_{BT} in the model) to apply a factor for system integration and test. Based on a review of the literature and actual data from 24 programs, the division of effort for a complete redesign task (see Figure 16) was established.

TASK	PERCENT OF TOTAL TASK
Analysis	15%
Program Design	20%
Code & Check	20%
Program Test	32%
System Test, Integration, & Documentation*	13%

*System test and integration as well as documentation are calculated separately.

Figure 16. Division of Effort for Complete Redesign Task

The division of effort was then used to calculate the R_{BC} rates shown in Figure 14.

Because R_{BC} and R_{BT} were based on development production rates and did not include certain aspects which are critical to program transfer, degradation factors which allow for these critical aspects were developed. These degradation factors impact total cost by adding to the total manpower requirements needed to transfer a program.

Three degradation factors were considered, (1) the quality and completeness of documentation, (2) the number and magnitude of modifications that will take place during the transfer (program instability), and (3) program complexity (system integration).

Degradation factors for documentation quality (D_{F1}) were based on intuitive judgment after much discussion. The factors recommended by the Hahn and Stone technique are shown in Figure 17. These factors are applied against R_{BC} for reprogram and recode.

DOCUMENTATION CATEGORY	% INCREASE IN MANPOWER
Excellent	0%
Good	25%
Average	50%
Poor	75%

Figure 17. Documentation Degradation Factors

The ease of program transfer is affected by the number and magnitude of modifications that will take place during the actual program transfer. Using historical data from System Development Corporation, Hahn and Stone calculated the degradation factors for program instability (D_{F2}) shown in Figure 18. These percentages are calculated against R_{BC} for redesign, reprogram, and recode.

MODIFICATIONS	% INCREASE IN MANPOWER
NIL	0%
TRIVIAL	5%
SOME	10%
EXTENSIVE	15%

Figure 18. Program Instability Degradation Factors

According to Hahn and Stone, program complexity will affect the amount of resources required to transfer the program. Using data provided by System Development Corporation, Hahn and Stone derived a degradation factor for system integration (D_{F3}) of $0.016X$, where X = number of subprograms* in a program. The results of this equation are used against R_{BT} .

In using the Hahn and Stone technique, the estimator starts off with program size in terms of instructions. Because the estimator is working with an existing program his estimate for size should be quite reliable. The technique is easy to use; the starting statistic (software size) is easy to get and various factors and rates need only be looked up and plugged into the equation.

In terms of accuracy, the model does not consider the problem of constraints on the target machine other than indirectly via a judgment of how difficult the target machine is to program. For this reason, it does not handle trade-offs directly. Further, Hahn and Stone do not mention any attempt to validate their model or their two basic assumptions; (1) the development production rate for computer programs follows a normal distribution, and (2) the program sample used is representative.

In terms of the model itself, Hahn and Stone state that:

$$C_T = C_P + C_D + C_0$$

where C_T = total cost

C_P = cost of individual program transfer

C_D = cost of transferring data

C_0 = other costs

*A computer program component or separately compilable module.

They further state that:

$$C_P = C_A + C_M$$

where: C_A = cost of machine-assisted transfer

C_M = cost of manual transfer

In deriving their model, it appears that C_A is somehow inadvertently dropped. Their final cost summary model is stated as follows:

$$C_T = \sum I \left[\frac{(1+D_{F3}) R_{BC} + (1+D_{F1} + D_{F2}) R_{BT}}{R_{BC} R_{BT}} \right] C_{MD} + C_D + \sum C_O$$

Manual Transfer Costs Data Costs Other Costs

1.2 PUTNAM GENERAL SOLUTION TO THE SOFTWARE SIZING AND ESTIMATING PROBLEM

The Putnam technique provides a methodology to produce life cycle estimates of total manpower and time required to reach critical milestones of software projects. It provides the means to allocate the cost among divisions of time during the software project life cycle rather than attempting to allocate the total cost among each part of the software project. (See Refs. [27, 28 and 29].)

1.2.1 Factors Considered in the Putnam Model

The basic model describing the software project has only two parameters: a magnitude parameter (total life cycle man-years) and a time parameter governing the shape of the curve. Since this model is only interested in man-years as a measure of work and total development time, it does not directly consider any constraints inherent in the development process.

1.2.2 Putnam Life Cycle Model

The Putnam technique is based upon a project profile taken from Norden (see Ref. [30]). Norden found that R&D projects are composed of man-power loading which can be linked to get a project profile. Figure 19 shows the individual cycles laid out in the expected time relationship. The sum of the underlined cycles is the man-power loading profile for the entire project, labeled Project Curve in Figure 19.

Putnam analyzed man-year budgetary data from the U.S. Army Computer Systems Command and concluded that the projects followed the life cycle model. He also concluded that the data fit the model with sufficient accuracy to provide a useful tool.

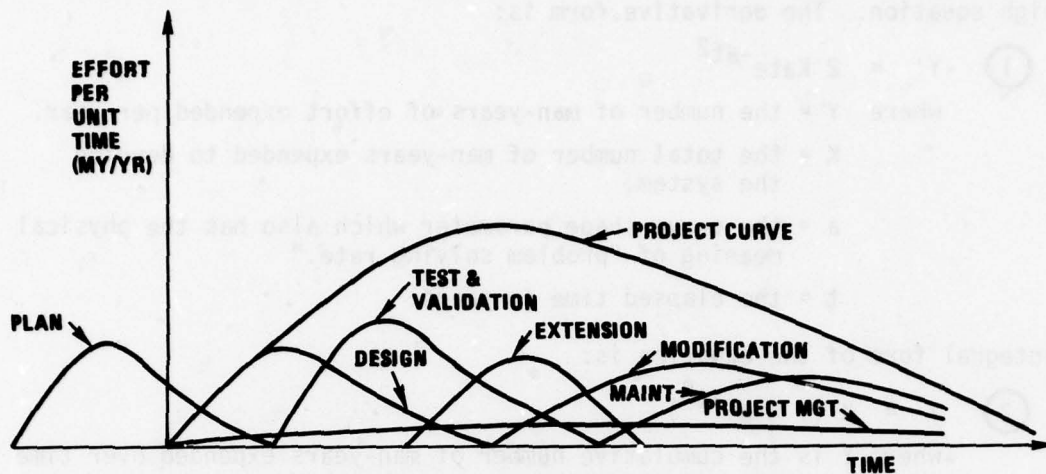


Figure 19. Project Profile

Figure 20 shows the typical Computer Systems Command (CSC) application of manpower to a software development project. The ordinates of the underlined cycles are added to obtain the total life cycle effort (or project curve) at various points in time.

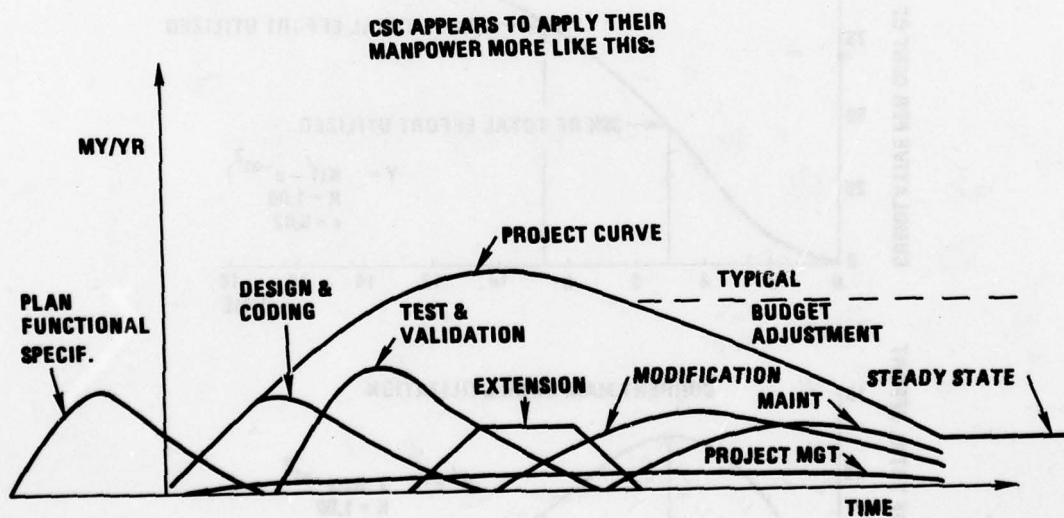


Figure 20. Typical CSC Application of Manpower to a Software Development Project

Putnam uses the Rayleigh equation, which has been empirically determined to fit the project man-power loading profile (Project Curve), to represent Norden's model. The model can be expressed in an integral and derivative form of the Rayleigh equation. The derivative form is:

$$\textcircled{1} \quad Y' = 2 K a t e^{-at^2}$$

where Y' = the number of man-years of effort expended per year.

K = the total number of man-years expended to develop the system.

a = the curve shape parameter which also has the physical meaning of "problem solving rate."

t = the elapsed time in years.

The integral form of the equation is:

$$\textcircled{2} \quad Y = K (1 - e^{-at^2})$$

where Y is the cumulative number of man-years expended over time t .

Figure 21 depicts both forms of the model. The derivative form (current man-power utilization) is used most frequently because budget data are in that form. The integral form (cumulative man-power utilization) is the typical S-shaped life cycle curve.

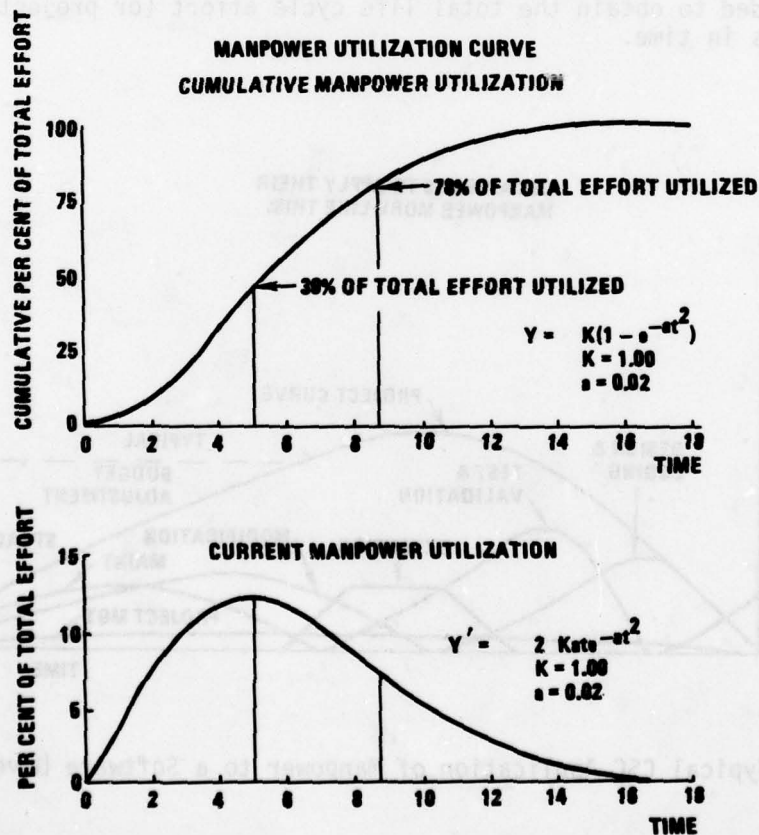


Figure 21. Life Cycle Integral and Derivative Curve Forms

Putnam also relates calendar milestones frequently established in connection with software development in the Life Cycle Model as shown in Figure 22. The milestones used by the Army are Design Certification, the Systems Integration Test (S.I.T.), the Prototype Evaluation Test (P.E.T.) and First Extension (Initial Operating Capability). These all occur on the rising part of the curve. Putnam notes that, First Extension occurs very close to the peak of the curve which has been empirically shown to very closely correspond to the development time (completion of design and coding as shown in Figure 22).

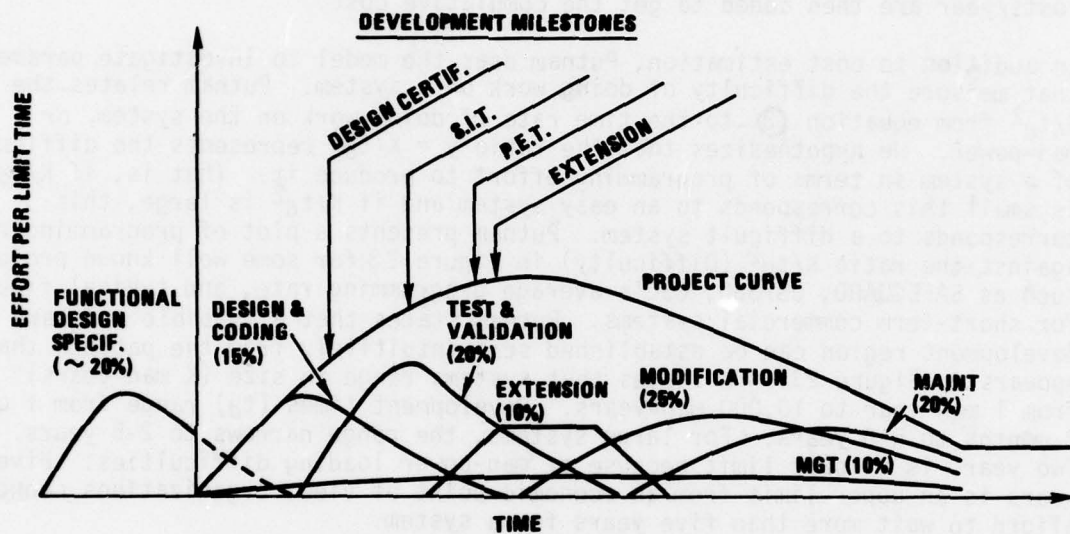


Figure 22. Application Software Life Cycle

Putnam states that the calendar milestones are empirically determined for the Army Computer Systems Command and that other design centers would differ to some extent. The general location should be similar for other design centers.

In equation (1), a can be replaced by the term $1/2t_d^2$, where t_d is the time for the curve to reach a maximum; resulting in the equation:

$$(3) \quad Y' = \frac{2K}{t_d^2} + T_e^{-t^2/2t_d^2}$$

Putnam states that t_d has been empirically shown to correspond very closely to the development time (completion of design and coding) of a large system. At time t_d , 39 percent of the total effort has been expended (see Figure 21).

Estimates of the two parameters of Putnam's model, K (the total life cycle man-years), and t_d (the time for the derivative curve to reach a maximum) can be used to derive an equation giving the ordinates of the man-power expenditure curve for a specific project. A yearly dollar costing can then be computed for the project by multiplying the ordinates of the man-power curve at each year by the average cost/man-year to arrive at a dollar cost/year. The dollar costs/year are then added to get the cumulative cost.

In addition to cost estimation, Putnam uses the model to investigate parameters that measure the difficulty of doing work on a system. Putnam relates the term K/t_d^2 from equation (3) to the time rate of doing work on the system, or man-power. He hypothesizes that the ratio $y = K/t_d^2$ represents the difficulty of a system in terms of programming effort to produce it. That is, if K/t_d^2 is small this corresponds to an easy system and if K/t_d^2 is large, this corresponds to a difficult system. Putnam presents a plot of programming rate against the ratio K/t_d^2 (Difficulty) in Figure 23 for some well known projects such as SAFEGUARD, OS/360, CSC's average programming rate, and typical figures for short-term commercial systems. Putnam states that a feasible software development region can be established semi-intuitively from the pattern that appears in Figure 23. He states that systems range in size (K man-years) from 1 man-year to 10,000 man-years. Development times (t_d) range from 1 or 2 months to 5-6 years. For large systems, the range narrows to 2-5 years. Two years is a lower limit because of man-power loading difficulties. Five years is an upper limit from an economic point of view; organizations cannot afford to wait more than five years for a system.

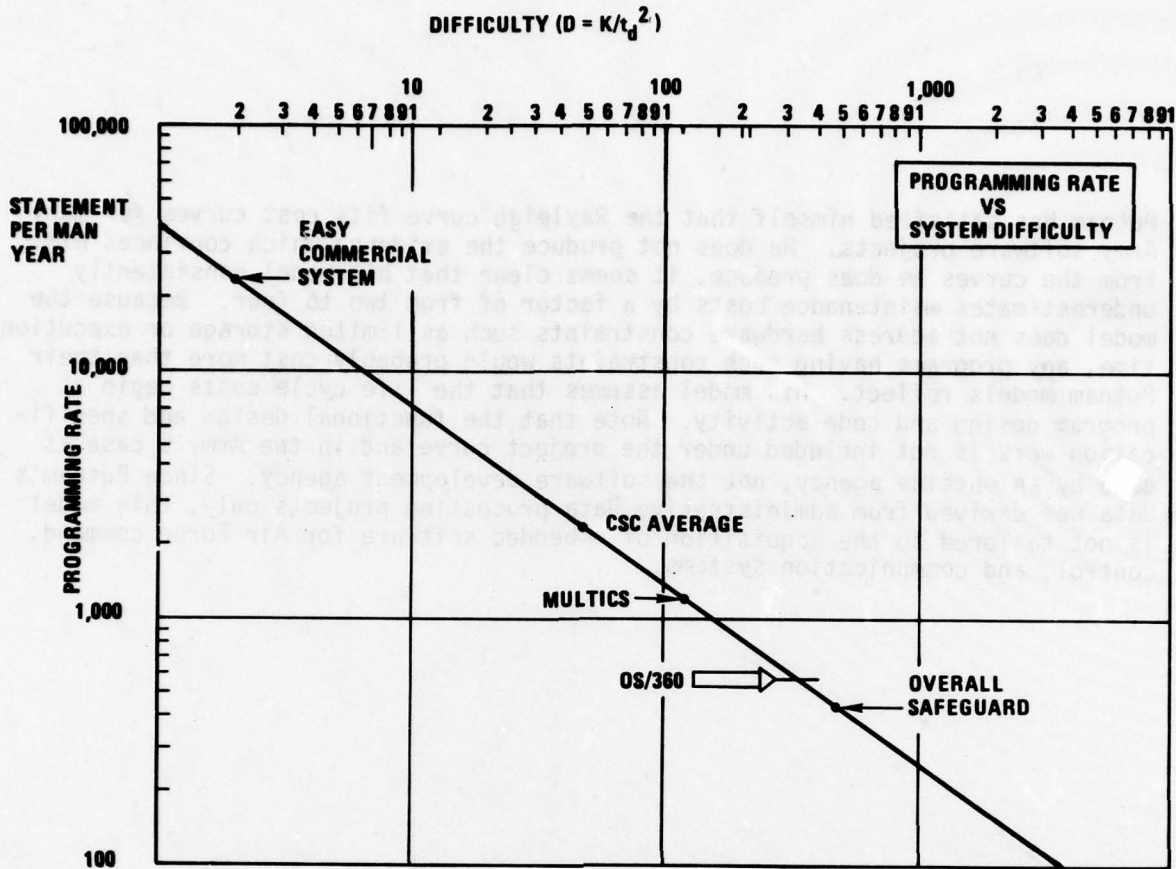


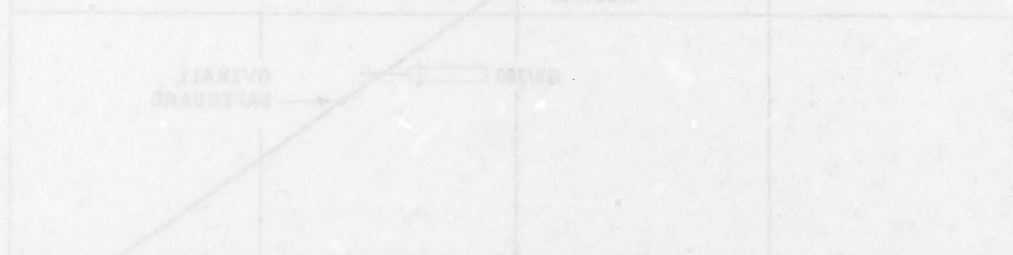
Figure 23. Programming Rate Versus Difficulty

1.2.3 Evaluation of the Putnam Model

The Putnam model is only interested in man-years as a measure of work. It does not attempt to split these man-years according to the type of work being done, nor does it try to estimate the amount of computer or other overhead needed. The only constraints of interest to the Putnam model are man-years and development time; other very real constraints are ignored. On the other hand, it does provide information, a breakdown of cost according to time, not normally available. It does not provide costs for non-manpower items such as computer time or travel.

The Putnam model is quite simple, almost simplistic. The amount of effort needed for an estimate is small because the estimation process can be automated easily. However the determination of K and t_d , based upon past history can be time consuming if such data is not readily available or in a readily usable form.

Putnam has satisfied himself that the Rayleigh curve fits cost curves for many Army software projects. He does not produce the evidence which convinced him. From the curves he does produce, it seems clear that his model consistently underestimates maintenance costs by a factor of from two to four. Because the model does not address hardware constraints such as limited storage or execution time, any programs having such constraints would probably cost more than their Putnam models reflect. His model assumes that the life cycle costs begin at program design and code activity. Note that the functional design and specification work is not included under the project curve and in the Army's case is done by an outside agency, not the software development agency. Since Putnam's data was derived from administrative data processing projects only, this model is not tailored to the acquisition of embedded software for Air Force command, control, and communication systems.



1.3 THE TECOLOTE PROVISIONAL MODEL FOR ESTIMATING COMPUTER PROGRAM DEVELOPMENT COSTS

The Tecolote model* is primarily concerned with military software developments in the context of larger weapon system hardware developments and emphasizes tactical software. The Tecolote model defines tactical software, which is time critical, as the complete set of computer programs that resides in and drives a computer system within a fire control system. A fire control system is further defined as "any set of military hardware which senses an enemy threat and then directs available resources against it." (See Ref. [29]).

The Tecolote model assumes that the computer hardware and a usable operating system already exists.

1.3.1 Factors Considered in the Tecolote Model

Seven basic elements were considered in the development of the Tecolote model, including:

- Threat characteristics
- Fast storage capacity
- Operational instructions
- Delivered instructions
- Development labor
- Computer costs
- Development costs

Two general types of threat characteristics are addressed in the model because of their effect on software development costs; the size of the threat and the speed of the threat. Tecolote defines size "as the maximum number of threat objects which must be simultaneously tracked during a terminal engagement." Speed is defined "as the maximum closing velocity manifested by any threat object."

For example, for a Navy fire control system like AEGIS, threat size is defined as the maximum number of air threats that must be intercepted simultaneously, and the threat speed is the maximum speed of any one of those air threats. Similarly, threat size in a Navy sonar system like BQQ5 is the maximum number of subsurface and surface threats that the ship has to counter simultaneously, and threat speed is the maximum speed of any one of those subsurface/surface threats.

*Derived for the Resource Analysis Branch, Code OP-96D, of the Office of the Chief of Naval Operations, Department of the Navy.

Fast storage capacity determines how much of the total information stored by a computer system can be retrieved and operated on at rates approximating the computer execution rate (the remainder of the information is stored on slow or bulk storage units where retrieval rates are on the order of a thousand times slower). In the case of tactical software, the requirement for fast storage capacity should increase as the number of targets to be tracked or the target approach speed increases.

The Tecolote model addresses software size in terms of operational and delivered instructions. Operational instructions are those procured during development that are eventually installed in the tactical hardware; delivered instructions are all those instructions produced during development, whether operational or not. (The instructions contained in a development test bed which simulates hardware interfaces are an example of delivered instructions which never become operational.)

Tecolote hypothesizes that as the number of targets to be tracked or the target approach speed increases, the number of operational instructions increases. And further, since a similar correspondence exists between fast storage capacity and target numbers and speed, a correlation exists between fast storage capacity used and the number of operating instructions, so long as target approach speed is held constant. However, as target approach speed increases, time-criticality increases and hence the fraction of the total operational program residing in fast storage should increase. Thus, the ratio of total operational instructions to require fast storage capacity should decrease as target speed increases.

According to Tecolote, in software developments which are part of large hardware developments, it is almost always true that the number of delivered instructions is greater than the number of operational instructions. There are always some hardware interfaces that do not exist at the time they are needed for software testing and, hence, must be simulated by the software engineers. The ratio of delivered instructions to operational instructions can thus be thought of as a kind of development "overhead," which should remain about constant in value, given the associated condition of a larger hardware development.

Tecolote defines development labor "as the total number of man-months of direct labor expended during the software development." They hypothesize "that development labor increases as the number of delivered instructions increases."

In any software development, significant amounts of computer time are used for debug and testing purposes. This time is usually charged on an hourly rental basis. Tecolote hypothesizes that the amount of computer time used increases as development labor increases, because computer time functions here as a tool used by the software engineers. The cost per computer hour depends on the computer used in development.

According to Tecolote, the total cost for software development is the cost for development labor (including direct salaries, overhead, and other loadings), plus computer costs (also loaded). Costs for development labor can be assumed to scale with the number of man-months. Thus, for a fixed development computer, total development costs should increase directly with the number of development man-months.

1.3.2 Tecolote's Provisional Cost Model

The Tecolote provisional model is based on data collected from five programs as shown in Figure 24. The first three programs are Navy tactical software developments. All three fit the following conditions:

- Development accomplished in the context of a larger hardware development.
- Little or no change in hardware specifications during development.
- Computer already developed and available.

The other two programs represent military software developments of unspecified application. Here, the applicable conditions are:

- No larger hardware development.
- Computer already developed and available.

All five represent programs begun within the last five years.

From the data in Figure 24, Tecolote organized and plotted the pairwise relationships hypothesized in the earlier discussion. These relationships are shown in Figures 25 through 28 and are summarized below.

Figure 25 shows the relationship between threat size (targets terminal-tracked), threat speed, and fast storage capacity required in tactical software packages. The three applicable data sets stratify into two classes with respect to threat speed -- air threats, with target approach speeds on the order of 250 meters/second or higher, and sea threats with target approach speeds of 50 meters/second or lower*. As expected, fast storage capacity increases with both the number of targets terminal-tracked and the target approach speed.

*Actual data have been omitted from Figure 25 for security reasons.

	TOTAL COST \$M, FY 73	TOTAL COMPUTER		TOTAL LABOR		TOTAL DELIVERED INSTRUCTIONS	TOTAL OPERATIONAL INSTRUCTIONS	TOTAL WORDS FAST STORAGE	DEVELOPMENT COMPUTER
		\$M	HOURS	\$M	MM				
CIWS					72		14,316	14,316	CDC 469
BQ05					1,985		192,000	64,000	UYK-7
AEGIS	39.60	3.03	39,450	36.57		1,090,000	776,000	227,000	UYK-7
TRW "CASE A"					570		102,400		CDC 3800
TRW "CASE B"					1,575		215,000		CDC 3800

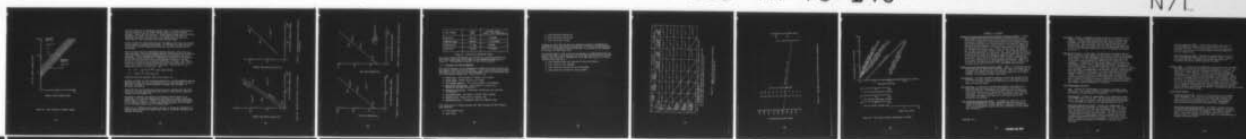
Figure 24. Basic Data for Tecolote Provisional Model

AD-A055 574 SYSTEM DEVELOPMENT CORP SANTA MONICA CALIF
SOFTWARE ACQUISITION MANAGEMENT GUIDEBOOK: COST EST.--ETC(U)
MAR 78 M FINFER, R MISH F19628-76-C-0236
UNCLASSIFIED SOC-TM-5772/007/02 ESD-TR-78-140

F/G 17/2

N/L

2 OF 2
AD
A055574



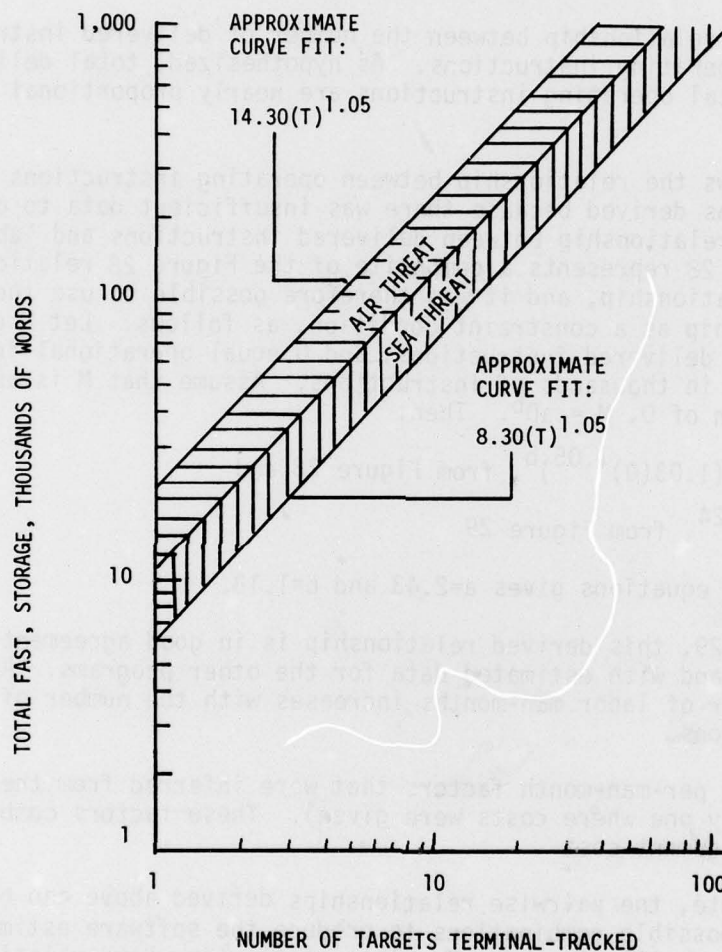


Figure 25. Fast Storage vs Targets Tracked

Figure 26 shows the correspondence between number of operating instructions, fast storage capacity, and target approach speed in tactical software packages. Again, just two classes of target approach speeds were considered, denoted by air threat and sea threat. As hypothesized, the number of operating instructions increases with fast storage capacity and, for a fixed capacity value, decreases with target approach speed.

Figure 27 shows the relationship between the number of delivered instructions and the number of operating instructions. As hypothesized, total delivered instructions and total operating instructions are nearly proportional to one another.

Figure 28 which shows the relationship between operating instructions and labor man-months, was derived because there was insufficient data to directly derive the desired relationship between delivered instructions and labor man-months. Figure 28 represents a composite of the Figure 28 relationship and the desired relationship, and it was therefore possible to use the Figure 27 relationship as a constraint condition, as follows: Let M equal man-months, D equal delivered instructions, and O equal operational instructions, both D and O in thousands of instructions. Assume that M is an exponential function of D, $M = aD^b$. Then:

$$M = aD^b = a(1.03(O)^{1.05})^b, \text{ from Figure 28 and}$$

$$M = 2.52(O)^{1.24}, \text{ from Figure 29}$$

Combining these two equations gives $a=2.43$ and $b=1.18$.

As shown in Figure 29, this derived relationship is in good agreement with the BQQ5 actual datum, and with estimated data for the other programs. As hypothesized, the number of labor man-months increases with the number of delivered instructions.

Figure 30 lists the per-man-month factors that were inferred from the AEGIS data point (the only one where costs were given). These factors combine to produce total development cost.

According to Tecolote, the pairwise relationships derived above can be composited in all possible combinations to produce the software estimating matrix shown in Figure 31. Further, Tecolote states that "the relationships in this matrix are useful for evaluating software proposals from the standpoint of software design as well as software cost."

Figure 32 is a nomograph which permits the user to assume any combination of cost-per-month and cost-per-computer-hour factors, and adjust the model cost estimate accordingly.

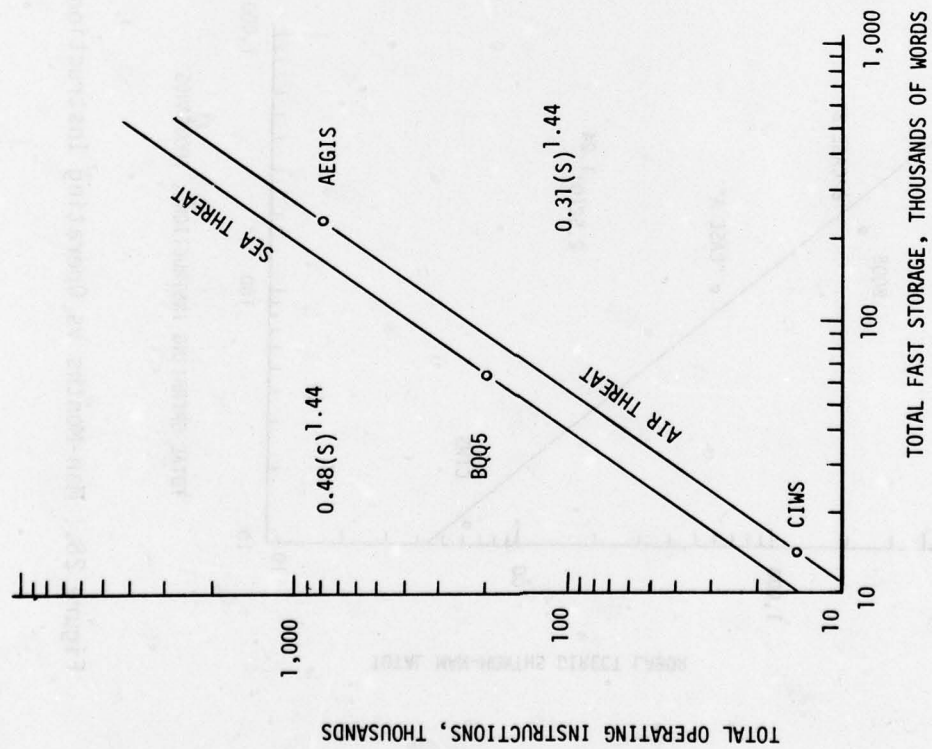


Figure 26. Operating Instructions vs Fast Storage

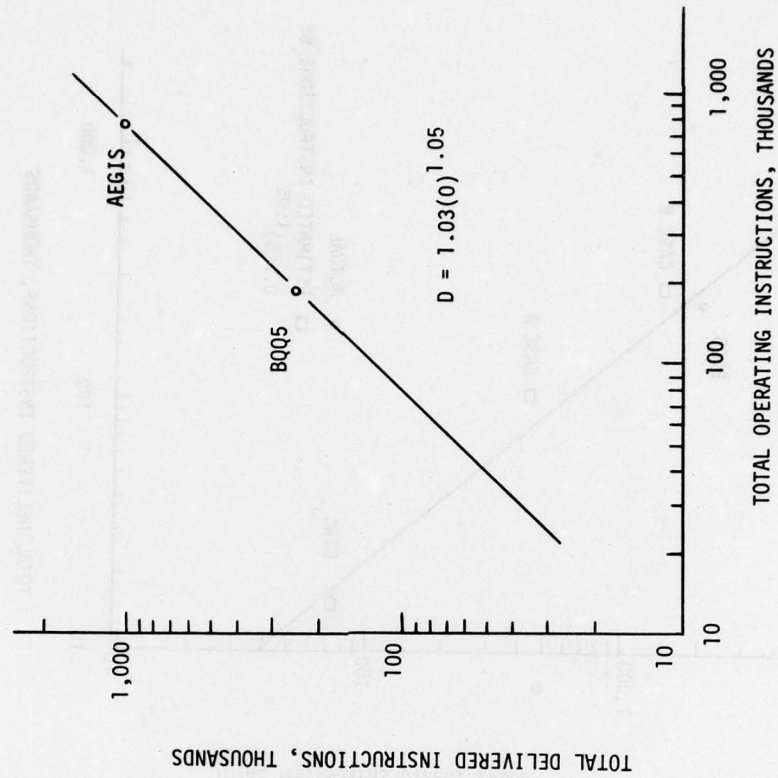


Figure 27. Delivered Instructions vs Operating Instructions

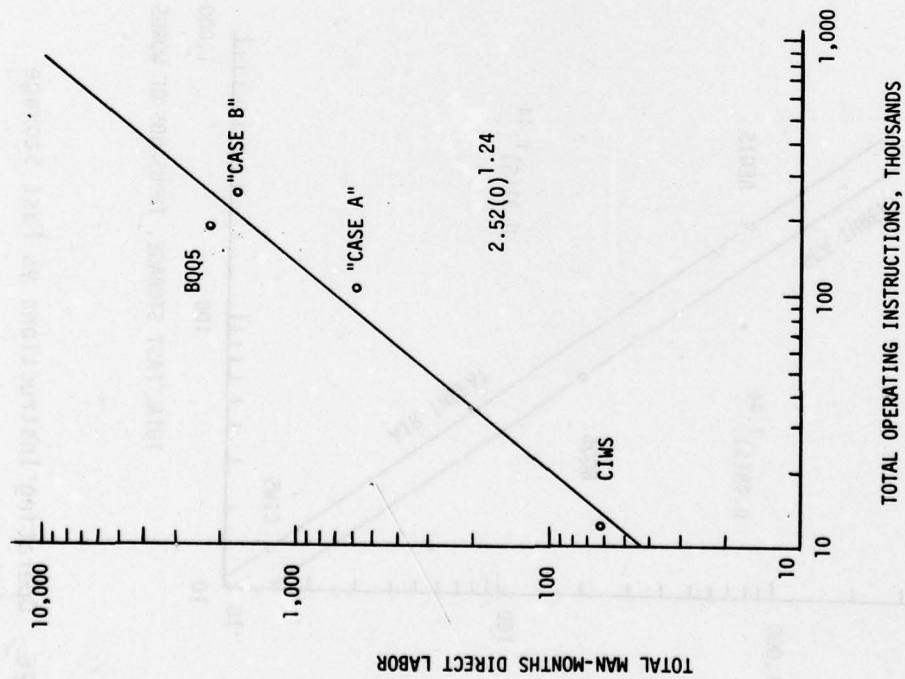


Figure 28. Man-Months vs Operating Instructions

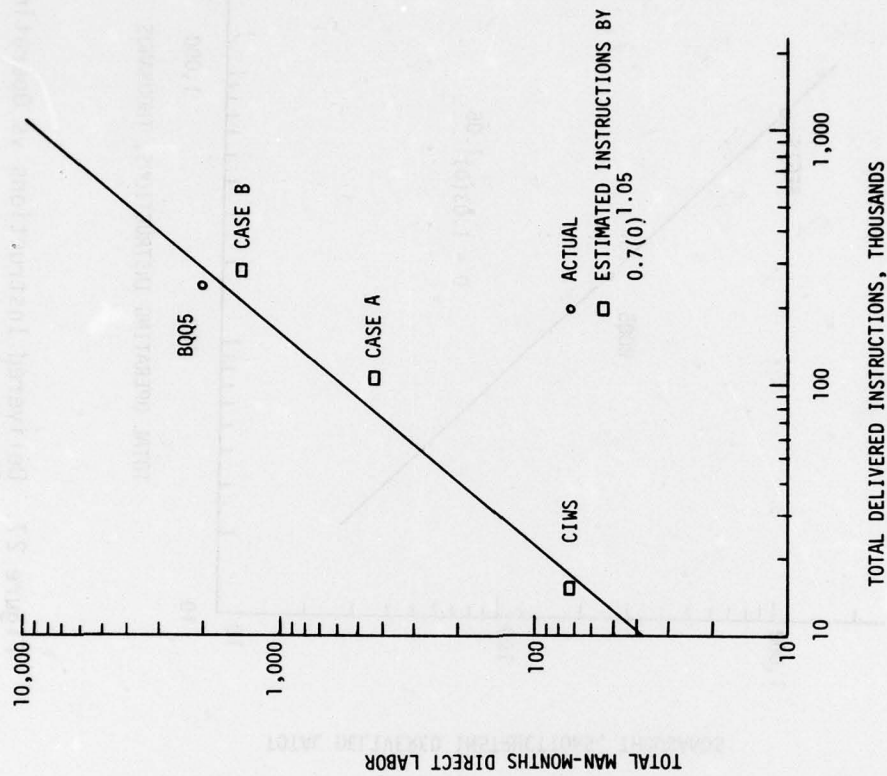


Figure 29. Man-Months vs Delivered Instructions

COST FACTORS	AEGIS TOTAL	INFERRED FACTOR (TOTAL ÷ 9,317 MAN-MONTHS)
LABOR COST	\$36.57M	\$3,930/MM
COMPUTER HOURS	\$39,450	4.23 HRS/MM
COMPUTER COST	\$ 3.03M	\$325/MM
TOTAL COST	\$39,50M	\$4,250/MM

Figure 30. Inferred per Man-Month Factors

The various software development cost estimating relationships derived for the Tecolote model (top row of Figure 31) are summarized graphically in Figure 33. The adjustment nomograph in Figure 32 also applies.

1.3.3 Evaluation of Tecolote Approach

The Tecolote approach uses an aggregated cost model to estimate costs for the development of military tactical software. Eight factors were selected which are known to impact tactical software development costs and hypothesized pairwise relationships between them. These factors are:

- Threat Size. Maximum number of targets to be tracked.
- Threat Speed. Maximum velocity of a target.
- Fast Storage Capacity. Size of main memory of the computer.
- Operational Instructions. Total number of installed (as opposed to test-bed) instructions.
- Delivered Instructions. Operational instructions plus test-bed instructions.
- Development Labor. Man-months of direct labor expended.
- Computer Costs. Money spent on computer time.
- Development Costs. Development labor plus computer costs.

Five large tactical software programs were then selected and the following data collected:

- Total computer hours
- Total labor

- Total delivered instructions
- Total operating instructions
- Total words fast storage

Subsequently, this data was used (via regression analysis) to demonstrate software estimating relationships between pairs of factors impacting development costs. These tend to show that the cost of such projects was dependent on the square of the threat size.

The Tecolote model is easy to use but its accuracy is indeed questionable because of the small size of the data base (five data points from five programs). Further, when Figure 24 is reviewed, it can be seen that only the following statistics were available:

- Total computer hours were obtained for only one program.
- Total labor for four programs.
- Total delivered instructions for two programs.
- Total words fast storage for three programs.

INPUTS OUTPUTS	M, TOTAL MAN-MONTHS LABOR	D, TOTAL DELIVERED INSTRUCTIONS (THOUSANDS)	O, TOTAL OPERATING INSTRUCTIONS (THOUSANDS)	AIR THREATS (B)		SEA THREATS (C)	
				S, TOTAL WORDS FAST STORAGE (THOUSANDS)	T, TARGETS TERMINAL- TRACKED	S, TOTAL WORDS FAST STORAGE (THOUSANDS)	T, TARGETS TERMINAL- TRACKED
TOTAL DEVELOPMENT COST FY 73 \$M	0.0043(M)	0.01(D) ^{1.18}	0.01(O) ^{1.24}	0.0026(S) ^{1.79}	0.30(T) ^{1.88}	0.0043(S) ^{1.79}	0.19(T) ^{1.88}
TOTAL MAN-MONTHS LABOR		2.43(D) ^{1.18}	2.52(O) ^{1.24}	0.59(S) ^{1.79}	69(T) ^{1.88}	1.01(S) ^{1.79}	45(T) ^{1.88}
TOTAL DELIVERED INSTRUCTIONS (THOUSANDS)			1.03(O) ^{1.05}	0.30(S) ^{1.51}	17(T) ^{1.59}	0.48(S) ^{1.51}	12(T) ^{1.59}
TOTAL OPERATING INSTRUCTIONS (THOUSANDS)				0.31(S) ^{1.44}	14(T) ^{1.51}	0.48(S) ^{1.44}	10(T) ^{1.51}
TOTAL WORDS FAST STORAGE (THOUSANDS)					14.30(T) ^{1.05}		8.30(T) ^{1.05}

NOTES: (A) COSTS ASSUME \$3,930/MM FOR LABOR, \$77/COMPUTER HOUR, AND 4.23 COMPUTER HOURS/MM. FOR OTHER FACTOR ASSUMPTIONS, SEE FIGURE 33.

(B) USE AIR THREAT COLUMN IF MAXIMUM THREAT APPROACH SPEED IS IN THE 250-700 M/SEC RANGE.

(C) USE SEA THREAT COLUMN IF MAXIMUM THREAT APPROACH SPEED IS 50 M/SEC OR LESS.

Figure 31. Summary of Provisional Software Estimating Relationships (See Note A)

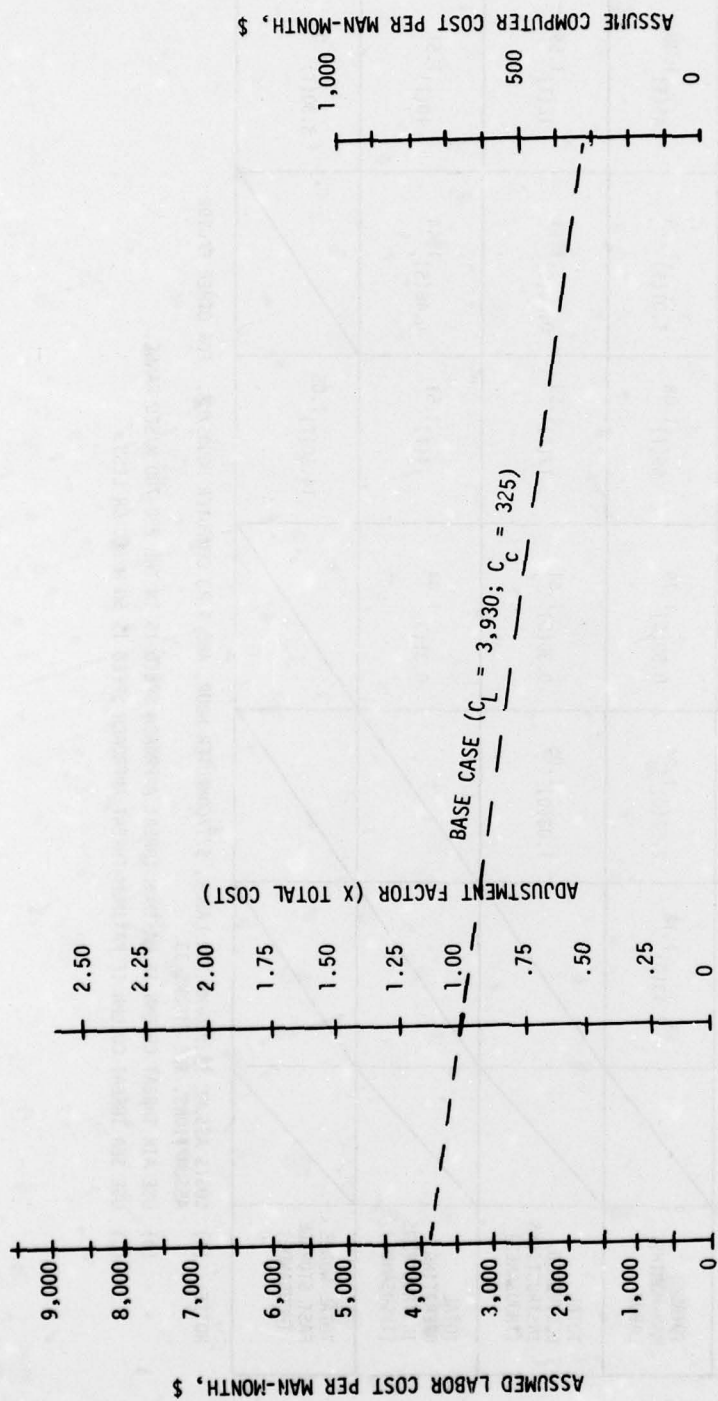


Figure 32. Adjustment Factor for Different Cost Factor Assumptions

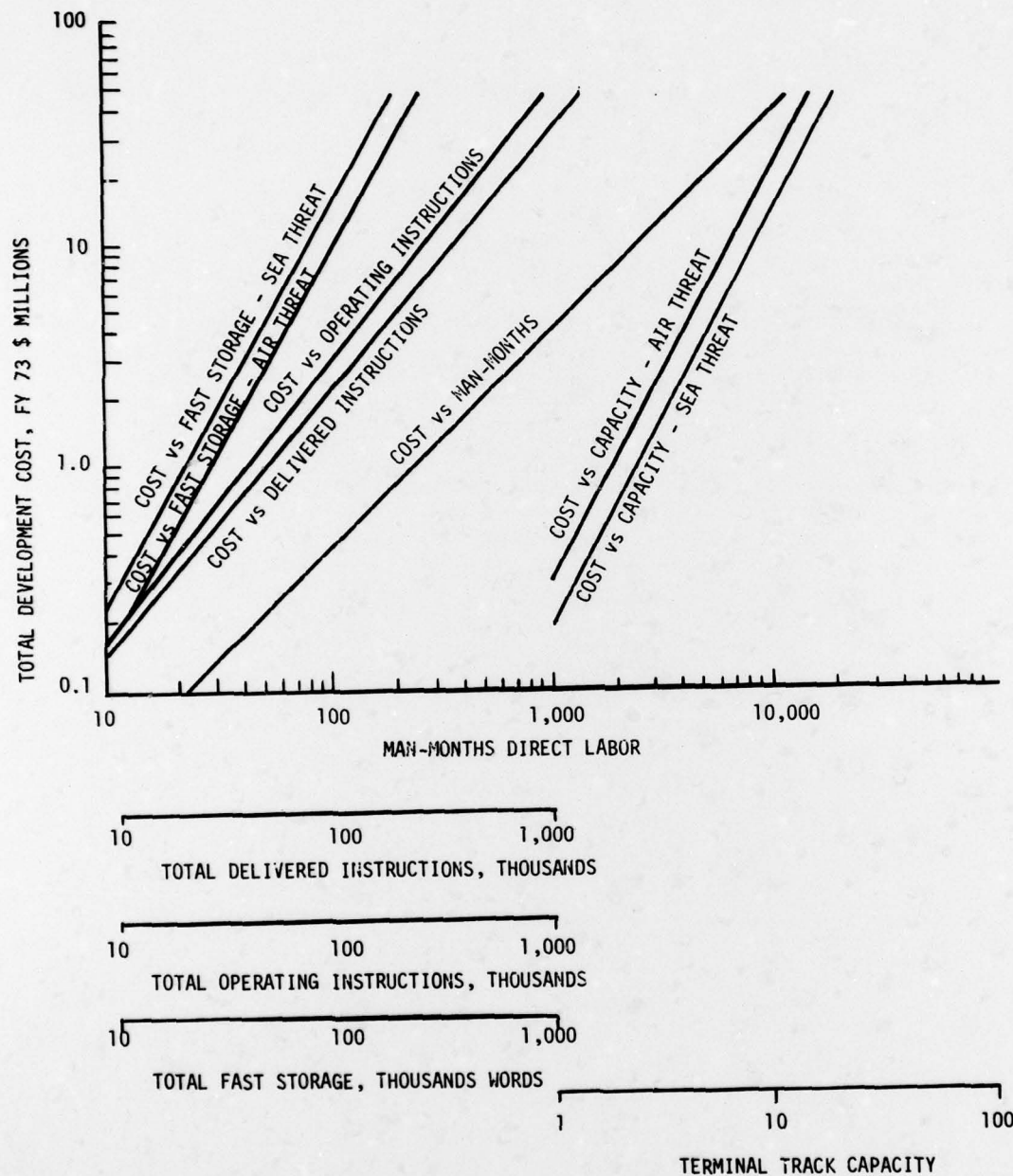


Figure 33. Provisional Software Development Cost Model

APPENDIX C - GLOSSARY

Computer-Assisted Design Specification Analysis Technique (CADSAT). A structured computer-aided system for specifying, recording, analyzing, and documenting design specifications for information processing systems. The system is composed of a user design specification language, a design specification analyzer, and a design specification reporting capability. The specification language is used to express a structured, unambiguous machine-processable form of all relevant requirements for the design of an information processing system. It has precise syntax and semantics, and includes non-scanable, textual material. The design specification is processed and stored in a data base. The design specification analyzer processes the data base and produces a number of reports that can be used by an analyst to verify that the design specification is complete and consistent. These reports can also be used to maintain the consistency of the specification as it is being produced and reviewed. CADSAT is a new name for the CARA system (see CARA) that emphasizes its application to design specifications as well as to requirements specifications.

Computer Assisted Requirements Analyzer (CARA). CARA is a structured, computer-aided technique for specifying, recording, analyzing, and documenting requirements for information processing systems. CARA's user requirements language, requirements analyzer, and reporting capability are virtually identical to its successor CADSAT (see CADSAT).

Cost Analysis. A process designed to examine or assess the validity of the estimated and actual resource requirements for a given program acquisition. Cost analysis activities are necessary for life cycle costs and design-to-cost evaluation.

Cost Data Base. A repository of software development cost elements (i.e., data) collected and maintained for the purpose of supporting cost estimation analyses. Requirements for establishing a cost data base must include consistency in item definitions, measures, and collection mechanisms. The data contained in a cost data base provide input for cost models and support cost analysis in the derivation of cost estimating relationships.

Cost Estimating Relationships (CERS). "A mathematical expression of the relationship between one or more independent variables (ordinarily stated in program terms, aircraft design, or physical characteristics, etc.) and the dependent variable (the cost attributable to the independent variables)."*

*From AFR 173-2.

Cost Model. Defines in mathematical terms the partial relationship of cost elements, software elements, development activities, resources, and schedules. A generalized software cost model applies specific data parameters to formulas derived from prior data analyses about cost relationships. Most cost models have been applied for the purpose of estimating software costs during the Full-Scale Development Phase and do not include Validation Phase activities or maintenance and support activities.

Cost-per-Instruction. A term used in the derivation of software cost estimates in which the total number of instructions contained in the software is divided by the total development cost to obtain a cost-per-instruction dollar figure. In using cost-per-instruction, the terms "cost" and "instruction" must be specifically defined and be consistent to be comparable. In addition, the term "total number of instructions" may be ambiguous. Often there is simultaneous development of support and application software. If the support software is not a deliverable, its size (in number of instructions) is absent from the total number of instructions delivered, and therefore is not recorded, although its development costs are recorded in total costs. A similar condition exists with frequent changes in requirements which cause code to be discarded. Further, it is very difficult to compare costs-per-instruction for different programs because there is no common basis for comparing the tasks, products, and services that are included in the cost-per-instruction figure. This guidebook clarifies some of the problems inherent in cost estimates and relationships of this nature.

Cost Reimbursement Contracts

- Cost. Provides for reimbursement of contractor's allowable costs, with no fee. This type of contract is usually used in research and development work with nonprofit institutions.
- Cost Sharing. Provides for reimbursement of an agreed upon portion of allowable costs, with no fee. The contractor bears part of the costs. This type of contract is used for projects jointly sponsored by the Government and industry with expected benefits to both.
- Cost Plus Incentive Fee (CPIF). Provides for reimbursement of allowable costs with provision for adjustment of fee in accordance with the relationship of final cost to estimated cost. At inception, maximum fee, minimum fee, and formula for sharing cost over and under the estimations are established. This type of contract is used primarily in development, where an estimated cost and fee formula can be negotiated that will provide the contractor with a positive incentive for effective management. This type of contract usually involves some amount of innovation in the work to be performed.

- Cost Plus Award Fee (CPAF). CPAF and CPIF contracts are similar. In both contracts the amount of fee is based on how well the contractor performs. In the case of CPAF, a board of review determines how the contractor is doing and awards a variable amount of fee over some base fee.
- Cost Plus Fixed Fee (CPFF). Provides for reimbursement of contractor's allowable costs and payment of a fixed fee. These contracts are usually awarded for research, preliminary exploration, or study where the level of effort required is unknown.

Design-to-Cost. A concept used to manage and control acquisition, operation, and support costs during the design and development of a system. The application of design-to-cost requires derivation of a specific cost target for a stated quantity, a schedule, and minimum performance requirements (i.e., selection of a unit-cost goal which becomes the principal design parameter in the development of the product). Design-to-cost is a concept utilizing unit-cost goals that represent what the Government has established as the price it is willing and able to pay for a unit of equipment or major system. Although the initial software cost estimates derived during the Conceptual Phase are grossly incorrect due to incomplete system definition, design-to-cost analysis requires initial software cost estimates for analysis of estimated system costs against known costs of existing systems.

Fixed Price Contracts

- Firm Fixed Price (FFP). Price is set initially and is not subject to any adjustment. The contractor assumes maximum financial risk, and all profits and all losses are his. This type of contract is used where prices are established at the outset. Requirements must be measurable and definite, requiring little innovation.
- Fixed Price with Escalation (FP-E). This type of contract provides for upward and downward revision of the stated contract price due to certain defined, measurable contingencies. This type of contract is used in cases where contract cost elements (such as labor rates, material costs, or component prices) are likely to be unstable over an extended performance period.

- Fixed Price Incentive Fee (FPIC). This type of contract provides for the adjustment of profit and contract price by a negotiated cost to target cost formula. The contractor may share in cost savings by higher profits, or may be penalized for overestimated costs, that can end in a loss. Other incentives may also be contractually specified which alter original cost estimates when savings are shared with the contractor.
- Fixed Price Incentive Firm (FPIF). At inception of FPIF, estimated costs, profit price ceiling, and formula for sharing costs over and under estimation are established. This type of contract is used when there is a modest degree of innovation and the contractor's assumption of a degree of cost responsibility will give him a positive profit incentive for effective cost control and contract performance.

Life Cycle Costs. Those costs, including direct, indirect, recurring, and nonrecurring costs, associated with a system's research, development, production, and deployment (operation and support) that are incurred as the total cost of ownership. Life cycle costing is a technique used to estimate and control a system's total costs, by the use of cost models. The components of life cycle costs need to be strictly defined. Estimates and cost reporting must adhere to the definitions so that life cycle cost comparisons between systems can be meaningful.

Life Cycle Cost (LCC) Models. Mathematical equations expressing the total Life Cycle Cost of a system, subsystem, or piece of equipment. The LCC model may include parameters for all costs incurred in the research, development, production, operation, and support of a system. LCC models generally have two functions: (1) the specification of the elements of cost which comprise the total life cycle costs, and (2) the relation of system design, performance, and deployment to the value of one or more LCC elements to estimate the cost impact of design alternatives.

Program Control. The management of program costs and schedules, including estimating, controlling, and the tracking and reporting of budgets, costs, schedules, and related management information associated with Air Force system acquisitions. The data associated with the management of each program's estimated/actual costs and schedules must be collected and maintained to provide information for future acquisitions.

Program Office (PO). The field office organized by the Air Force manager responsible for an assigned program. Its purpose is to assist him in the development, testing, and procurement of systems, subsystems, equipment, modifications, supporting projects, and studies.

Software Cost Estimation. Software cost estimation may be defined as the process of predicting the cost of resources needed to complete a set of activities which result in delivery of a software product or set of software-related products. The estimation process is an essential activity in the acquisition of any software product.

Software Cost Measurement. Software cost measurement may be defined as the process of appraising the expenditure of resources allocated for the set of activities required to produce a given software product. Software cost estimation and measurement are two functions associated with the discipline of program control.

Software Requirements Engineering Methodology (SREM). A part of the Ballistic Missile Defense Advanced Technology Center (BMDATC) program directed toward improving the methodology used in the development of software for BMD programs. SREM is a part of this program and addresses requirements generation for software development. SREM is composed of a combination of languages, tools, and procedures designed to reduce or eliminate known error sources. Data processing subsystem requirements in the SREM approach for BMD are predicated on an input-processing-output orientation with processing flow through the subsystem being described in terms of required paths through the subsystem. Each processing path is identified by a message (stimulus), a sequence of processing steps (including required decision nodes with decision variables), and response with data which are local to processing on a path, or which must be saved for processing on a subsequent path being processed. Performance requirements in SREM are defined, at a minimum, in terms of the accuracy and timing required across the subsystem for each stimulus and are stated on each path or sequence of paths in the form of validation points. These identified points mark places in the processing where specified data must be available for collection, and executable procedures which define the performance pass-fail criteria that will be imposed on the software.

Specification and Assertion Language (SPECIAL). A design specification language developed in conjunction with the Stanford Research Institute (SRI) design, implementation, and formal verification methodology, SRI Hierarchical Design Methodology (HDM). SPECIAL was designed to (1) specify systems constrained within a particular methodology (HDM) for the design, implementation, and proof of computer systems; (2) specify systems containing hardware and software; (3) permit syntactic checks on the consistency of a specification; (4) be directly usable in a formal proof of correctness; and (5) specify systems that can be implemented in any programming language. SPECIAL is a design specification language based on a mathematical discipline. It constrains the system design to conform to HDM for structuring and implementing systems. While these are desirable properties of SPECIAL, they also require specialized training and dedication to the specific methodology.

APPENDIX D - OFFICIAL GOVERNMENT DOCUMENTS FOR COST ESTIMATION AND MEASUREMENT

DODI 4105.62, Proposal Evaluation and Source Selection
DODI 5000.19-L, Acquisition Management Systems and Data Requirements,
Control List
DODI 5000.22, Guide to Estimating Costs of Information Requirements
DODI 7000.2, Performance Measurement for Selected Acquisitions
DODI 7000.8, Cost Performance Report
DODI 7000.10, Contract Cost Performance Funds Status and Cost Schedule
Status Reports
DODI 7000.11, Contractor Cost Data Reporting
DODI 7041.3, Economic Analysis and Program Evaluation
DODI 7220.25, Standard Rates for Costing Military Personnel Services
LCC-1, Life Cycle Costing Procurement Guide
LCC-3, Life Cycle Costing Guide for System Acquisition
MIL-STD-881A, Work Breakdown Structure for Defense Materiel Items
MIL-STD-1641, Preparation of Pert/Time Networks and Reports for Training
Device Contracts
MIL-P-23189A (Navy), PERT/Time and PERT/Cost Management Information Systems
for Planning and Control
AFM 70-6, Source Selection Procedures
AFP 70-14, PIECOST (Probability of Incurring Estimated Cost)
AFR 173-1, Management of the Cost Analysis Program
AFR 173-2, USAF Cost Estimating Relationship/Cost Factors Program
AFM 173-10, USAF Cost and Planning Factors
AFM 173-11, Independent Cost Analysis Program
AFR-174-2, Follow-Up on Internal Reports of Audit (AFSC supplement 11/2/72
and ESD supplement 6/15/72)
AFR 175-4, Auditing in the Air Force
AFM 175-118, Air Force Audit/Management System
AFR 177-13, Accounting for Research and Development
AFM 177-100, General Principles, Standards, and Policies of the Air Force
Accounting and Finance System
AFR 178-1, Economic Analysis and Program Evaluation for Resource Management
AFR 310-1, Management of Contractor Data

AFR 600-1, Development, Selection and Application of Management Control Systems
 AFR 800-6, Program Control - Financial (AFSC supplement 9/4/74)
 AFR 800-11, Life Cycle Costing (LCC)
 AFR 800-14, Management of Computer Resources in Systems
 AFSCP 27-1, Program Direction
 AFSCM 173-1, Cost Estimating
 AFSCM 173-2, Cost Information System
 AFSCP 173-5, Cost/Schedule Control Systems Criteria
 AFSCP 173-6, C/SCSC Joint Surveillance Guide
 AFSCR 310-1, Management of Contractor Data
 AFSCR 310-2, Deferred Requisitioning of Engineering Data
 AFSCP 800-2, Management of Multi-Service Systems, Programs, Projects
 AFSCP 800-3, A Guide to Program Management
 AFSCP 800-6, Statement of Work Preparation Guide
 AFSCP 800-14, Joint AMC/NMC/AFLC/AFSC List of Authorized Acquisition Management Systems
 AFSCP 800-15, Contractor Cost Data Reporting (CCDR) System
 AFSCP 800-19, Joint Design-to-Cost Guide: A Conceptual Approach for Major Weapon System Acquisition
 AFSCP 800-23, Secretary of the Air Force Program Review/ Program Assessment Review/Command Assessment Review (SPR/PAR/CAR) Guidance
 ASPR, Armed Service Procurement Regulation
 ESDR 173-1, Electronic Systems Division Cost Analysis Program
 ESDR 173-3, Cost/Schedule Control System Criteria (C/SCSC)
 AFR 375-8, Work Breakdown Structure for Defense Materiel Items
 AFSCM 173-4, Program Breakdown Structure and Codes
 ESDR 173-2, Work Breakdown Structures

COMMENT SHEET

Software Cost Estimation and Measurement Guidebook

Reviewer's Name:

Reviewer's Organization:

Comments:

(11/15)

(11/15)

THOT/122 PH

28 2028

12/10 AM 1/15 12/10/15

Please return to: HQ ESD/TOIT (Stop 36)
Hanscom AFB, MA 01731

SECRET

Software Cost Estimation and Measurement Guidelines

Software's Organization

Software's Name

Comments

(FOLD)

(FOLD)

FROM:

HQ ESD/TOIT

Stop 36

Hanscom AFB, MA 01731

116